IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Appl. No.	:	09/748,716	Confirmation No. 5358
Applicant	:	Sara Elo DEAN et al.	
Filed	:	December 22, 2000	
TC/A.U.	:	2173	
Examiner	:	Brian J. DETWILER	
Docket No.	:	POUG920000205US1	
Customer No.	:	23334	

RED LINE

37 C.F.R. 1.131 DECLARATION

I, each and every one of the undersigned inventors of the above-referenced patent application, hereby declare the following:

1) Claims 1-9, 11-31, and 33-39 in our above-identified patent application were rejected under 35 U.S.C. §102(e) and claims 10 and 32 were rejected under 35 U.S.C. § 103(a) based on U.S. Patent Publication No. 2002/0085020 A1 to Carroll, Jr., entitled "XML-Based Graphical User Interface Application Development Toolkit" filed on September 14, 2001, with a priority date of September 14, 2000 ("Carroll").

2) The invention described in the above-referenced patent application was reduced to a writing prior to the September 14, 2000 priority date of Carroll. In particular, *Franklin Content Management Prototype* documentation (exhibit A), upon which the above referenced patent application was based, is attached herewith. The documentation is a comprehensive specification and installation of the inventive system (see the table of contents of this document for the full detail) created and used by the inventors prior to the September 14, 2000 priority date of Carroll and demonstrating features of the presently claimed invention. It includes everything from an Installation guide, configuration, setup of the DB and a Franklin workspace for content management, setting up of users, roles, and includes code snippets of communication between components and error codes.

3) Additionally, the invention described in the above-referenced patent application was reduced to actual practice prior to the September 14, 2000 priority date of Carroll. Proof of actual reduction to practice upon which the presently claimed invention was based is attached herewith and will be described in detail below.

4) Submitted herewith as evidence of actual reduction to practice prior to the September 14, 2000 priority date of Carroll are the following exhibits:

Exhibit A) In particular, *Franklin Content Management Prototype* documentation (exhibit A), upon which the above referenced patent application was based, is attached herewith. (PDF File Pages 1-50)

Exhibit B) Assignments passed out to users prior to the September 14, 2000 priority date of Carroll to test users who were evaluating the

integration between two systems: the present invention and "Kittyhawk" prior to the September 14, 2000 priority date of Carroll. The scenarios ask users to do different actions in the present invention's UI, which would show that there was a running system that could support users prior to the September 14, 2000 priority date of Carroll. The document describes the integration of the two systems, and shows the request/responses part of the communication between the two systems. (PDF File Pages 51-70)

Exhibit C) A copy of a State chart of the invention's DB with each possible state of a fragment when stored in the invention's DB. The State chart was created and used by the inventors prior to the September 14, 2000 priority date of Carroll and demonstrates features of the presently claimed invention. (PDF File Pages 70-71)

Exhibit D) Copies of HTML pages created by the inventors prior to the September 14, 2000 priority date of Carroll and demonstrating features of the presently claimed invention. The HTML pages describe to users how to install the inventive client and issue commands to manage documents, such as Check in, Check out, review, publish and describes the fragment/servable relationship to users. (PDF File Pages 72-75)

Exhibit E) A synthesis of all feedback from a user acceptance testing of the Invention, run prior to the September 14, 2000 priority date of Carroll. It includes a list of things users liked and did not like, which evidences that users were using the running end-to-end inventive system with features of the presently claimed invention prior to the September 14, 2000 priority date of Carroll. (PDF File Pages 76-90)

Exhibit F) A copy of brief notes identified during a code review of the invention's server code made prior to the September 14, 2000 priority date of Carroll. (PDF File Page 91)

Exhibit G) An email correspondence to persons other than the inventors of the present invention, listing the internet address for accessing, and instructions on how to use, the working prototype system created and used by the inventors prior to September 14, 2000 priority date of Carroll and demonstrating features of the presently claimed invention. (PDF File Pages 92-94)

Exhibit H) An email correspondence with reviewer feedback on the working prototype system created and used by the inventors prior to the September 14, 2000 priority date of Carroll and demonstrating features of the presently claimed invention. (PDF File Pages 95-98)

Exhibit I) Copies of several screenshots of the working prototype system created and used by the inventors prior to September 14, 2000 priority date of Carroll and demonstrating features of the presently claimed invention. These screenshots show lists of XML documents having content objects and content fragments which are named and linked through the entry fields. (PDF File Pages 99-101)

Exhibit J) A copy of a section of the source code file that was created and used by the inventors prior to September 14, 2000 priority date of Carroll and that implemented part of a working prototype system that performed features of the presently claimed invention. (PDF File Pages 102-103)

Exhibit K) A copy of the source code FranklinEditor.java that was created and used by the inventors prior to September 14, 2000 priority date of Carroll and that implemented part of a working prototype system that performed features of the presently claimed invention. (PDF File Pages 1-31)

Exhibit L) A copy of the source code InterfaceMaker.java that was created and used by the inventors prior to September 14, 2000 priority date of Carroll and that implemented part of a working prototype system that performed features of the presently claimed invention. (PDF File Pages 1-16)

5) The evidence submitted herewith supports the reduction to practice. The following table is submitted to show how each claim element is supported and that the test results unequivocally establish this software existed and worked for its intended purpose.

Claim1 is an example. The other independent claims (18, 23, & 39) recite identical limitations.

Claim 1: A method on an information processing unit for performing steps for assembling, with a user interface (UI), a document that conforms to a particular document type definition, the method comprising:

receiving a user selection for a document type	Exhibit B, page 3 under Scenario I, step 2.3, describes the step of creating the appropriate fragment: *"Start the task to create the appropriate fragment, fill it in, and check it in."* Exhibit A, page 25 "Create new content" paragraph shows the process of creating new content. Here the user must select a document type from the **File > New Fragment** menu or the **File > New Page** menu. These menus list all the available DTD types and it generates a template, for display in the UI, from the DTD selected. Exhibit A, page 25 "Create new content" and Exhibit D, page 2 "Create New Fragment" and

	"Create New Page" sections describe the procedure that a user has to perform to select and create a new document abiding to a document type. The system point of view of this process is the "reception of a user selection for a document type". Exhibit K, line 1095 Function **getToolBarPane** shows creation of UI to allow user to select a "new fragment or page" Lines 1099-1105 creates the button to do the action. e.g., iv_newButton.setToolTipText ("Create new fragment or page"); Exhibit K at line 760 Function **getFragmentTypeMenu** produces the menu for the user to select the type of DTD. Comment at 752-758 describes the function
selecting one of a plurality of document type definition types based upon the document type received;	~~Exhibit A, pages 9-10 is the definition process of a typical DTD; step 6 refers to attributes of user input needed, e.g. "string" or "longtext"; and Exhibit A, page 13-14, shows an example of a servable DTD.~~Exhibit A, page 25, "Create new content" Here we are selecting a DTD based on the user selection of a document type. When the user has selected a type of document to create from the menu, the system retrieves the correct DTD from the "**appropriate URL**" and then generates the UI (template). Each URL represents a different DTD to use. Exhibit A, page 25 "Create new content" and Exhibit D, page 2 "Create New Fragment" and "Create New Page" sections. The user is presented with a list of types and selects one. The system selects the document type definition (DTD) corresponding to the chosen document type. Exhibit K at line 260-270 **createFragment**(String lv_name) creates the appropriate fragment from the user selection. Comment before function (248-259) describes function action.
parsing one or more of a plurality of	Exhibit A, pages 9-10 shows the plurality of

elements in the document type definition type selected;	elements in a DTD~~. Pages~~ in the Franklin specification of fragment and servable DTDs. Further Exhibit A, pages 9-10 refer to the UI types, i.e., requirements for user input and ~~Pages~~Exhibit A, pages 13-14 show an example of a servable DTD. Exhibit A, page 25 "Editor UI Widgets" the DTD is parsed and based on the DATATYPE generates the UI widget. Here we see the mapping from DATATYPE to java widget (e.g., string => JtextField) Exhibit A, pages 13-14 shows an example of a servable DTD and the set of elements that makes up that servable DTD. The system parses this plurality of elements to create the user interface; an example of such a user interface is shown on the right panel of Exhibit I, page 2 under <IMAGEFRAGEMENT 3: title> Exhibit L, lines 114-134) "/** * For the given DTD and content model node, create appropriate input widgets and add to the JPanel. ...." this comment indicates the function **createInterfaceForModel** creates the widgets based on the DTD definition and the elements of the content. Describes use of DATATYPE to select the widget.

We, the undersigned, declare all of the above statements are made on our own knowledge, the above statements are true and correct, and the above statements are made on information that we believe to be true. We understand that false statements or concealment in obtaining a patent will subject us to fine and/or imprisonment or both (18 U.S.C. §1001) and may jeopardize the validity of the above identified patent application or any application issuing therefrom.

Louis WEITZMAN	Sara ELO DEAN	Dikran S. MELIKSETIAN
~~May~~October ___, 2006	~~May~~October ___, 2006	~~May~~October ___, 2006

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Appl. No.	:	09/748,716	Confirmation No. 5358
Applicant	:	Sara Elo DEAN et al.	
Filed	:	December 22, 2000	
TC/A.U.	:	2173	
Examiner	:	Brian J. DETWILER	
Docket No.	:	POUG920000205US1	
Customer No.	:	23334	

37 C.F.R. 1.131 DECLARATION

I, each and every one of the undersigned inventors of the above-referenced patent application, hereby declare the following:

1) Claims 1-9, 11-31, and 33-39 in our above-identified patent application were rejected under 35 U.S.C. §102(e) and claims 10 and 32 were rejected under 35 U.S.C. § 103(a) based on U.S. Patent Publication No. 2002/0085020 A1 to Carroll, Jr., entitled "XML-Based Graphical User Interface Application Development Toolkit" filed on September 14, 2001, with a priority date of September 14, 2000 ("Carroll").

2) The invention described in the above-referenced patent application was reduced to a writing prior to the September 14, 2000 priority date of Carroll. In particular, *Franklin Content Management Prototype* documentation (exhibit A), upon which the above referenced patent application was based, is attached herewith. The documentation is a comprehensive specification and installation of the inventive system (see the table of contents of this document for the full detail) created and used by the inventors prior to the September 14, 2000 priority date of Carroll and demonstrating features of the presently claimed invention. It includes everything from an Installation guide, configuration, setup of the DB and a Franklin workspace for content management, setting up of users, roles, and includes code snippets of communication between components and error codes.

3) Additionally, the invention described in the above-referenced patent application was reduced to actual practice prior to the September 14, 2000 priority date of Carroll. Proof of actual reduction to practice upon which the presently claimed invention was based is attached herewith and will be described in detail below.

4) Submitted herewith as evidence of actual reduction to practice prior to the September 14, 2000 priority date of Carroll are the following exhibits:

Exhibit A) In particular, *Franklin Content Management Prototype* documentation (exhibit A), upon which the above referenced patent application was based, is attached herewith. (PDF File Pages 1-50)

Exhibit B) Assignments passed out to users prior to the September 14, 2000

priority date of Carroll to test users who were evaluating the integration between two systems: the present invention and "Kittyhawk" prior to the September 14, 2000 priority date of Carroll. The scenarios ask users to do different actions in the present invention's UI, which would show that there was a running system that could support users prior to the September 14, 2000 priority date of Carroll. The document describes the integration of the two systems, and shows the request/responses part of the communication between the two systems. (PDF File Pages 51-70)

Exhibit C) A copy of a State chart of the invention's DB with each possible state of a fragment when stored in the invention's DB. The State chart was created and used by the inventors prior to the September 14, 2000 priority date of Carroll and demonstrates features of the presently claimed invention. (PDF File Pages 70-71)

Exhibit D) Copies of HTML pages created by the inventors prior to the September 14, 2000 priority date of Carroll and demonstrating features of the presently claimed invention. The HTML pages describe to users how to install the inventive client and issue commands to manage documents, such as Check in, Check out, review, publish and describes the fragment/servable relationship to users. (PDF File Pages 72-75)

Exhibit E) A synthesis of all feedback from a user acceptance testing of the invention, run prior to the September 14, 2000 priority date of Carroll. It includes a list of things users liked and did not like, which evidences that users were using the running end-to-end inventive system with features of the presently claimed invention prior to the September 14, 2000 priority date of Carroll. (PDF File Pages 76-90)

Exhibit F) A copy of brief notes identified during a code review of the invention's server code made prior to the September 14, 2000 priority date of Carroll. (PDF File Page 91)

Exhibit G) An email correspondence to persons other than the inventors of the present invention, listing the internet address for accessing, and instructions on how to use, the working prototype system created and used by the inventors prior to September 14, 2000 priority date of Carroll and demonstrating features of the presently claimed invention. (PDF File Pages 92-94)

Exhibit H) An email correspondence with reviewer feedback on the working prototype system created and used by the inventors prior to the September 14, 2000 priority date of Carroll and demonstrating features of the presently claimed invention. (PDF File Pages 95-98)

Exhibit I) Copies of several screenshots of the working prototype system created and used by the inventors prior to September 14, 2000 priority date of Carroll and demonstrating features of the presently claimed invention. These screenshots show lists of XML documents having content objects and content fragments which are

named and linked through the entry fields. (PDF File Pages 99-101)

Exhibit J) A copy of a section of the source code file that was created and used by the inventors prior to September 14, 2000 priority date of Carroll and that implemented part of a working prototype system that performed features of the presently claimed invention. (PDF File Pages 102-103)

Exhibit K) A copy of the source code FranklinEditor.java that was created and used by the inventors prior to September 14, 2000 priority date of Carroll and that implemented part of a working prototype system that performed features of the presently claimed invention. (PDF File Pages 1-31)

Exhibit L) A copy of the source code InterfaceMaker.java that was created and used by the inventors prior to September 14, 2000 priority date of Carroll and that implemented part of a working prototype system that performed features of the presently claimed invention. (PDF File Pages 1-16)

5) The evidence submitted herewith supports the reduction to practice. The following table is submitted to show how each claim element is supported and that the test results unequivocally establish this software existed and worked for its intended purpose.

Claim1 is an example. The other independent claims (18, 23, & 39) recite identical limitations.

Claim 1: A method on an information processing unit for performing steps for assembling, with a user interface (UI), a document that conforms to a particular document type definition, the method comprising:

receiving a user selection for a document type	Exhibit B, page 3 under Scenario I, step 2.3, describes the step of creating the appropriate fragment "*Start the task to create the appropriate fragment, fill it in, and check it in.*" Exhibit A, page 25 "Create new content" paragraph shows the process of creating new content. Here the user must select a document type from the **File > New Fragment** menu or the **File > New Page** menu. These menus list all the available DTD types and it generates a template, for display in the UI, from the DTD selected. Exhibit A, page 25 "Create new content" and

	Exhibit D, page 2 "Create New Fragment" and "Create New Page" sections describe the procedure that a user has to perform to select and create a new document abiding to a document type. The system point of view of this process is the "reception of a user selection for a document type". Exhibit K, line 1095 Function **getToolBarPane** shows creation of UI to allow user to select a "new fragment or page" Lines 1099-1105 creates the button to do the action. e.g., iv_newButton.setToolTipText ("Create new fragment or page"); Exhibit K at line 760 Function **getFragmentTypeMenu** produces the menu for the user to select the type of DTD. Comment at 752-758 describes the function
selecting one of a plurality of document type definition types based upon the document type received;	Exhibit A, page 25, "Create new content" Here we are selecting a DTD based on the user selection of a document type. When the user has selected a type of document to create from the menu, the system retrieves the correct DTD from the "**appropriate URL**" and then generates the UI (template). Each URL represents a different DTD to use. Exhibit A, page 25 "Create new content" and Exhibit D, page 2 "Create New Fragment" and "Create New Page" sections. The user is presented with a list of types and selects one. The system selects the document type definition (DTD) corresponding to the chosen document type. Exhibit K at line 260-270 **createFragment**(String iv_name) creates the appropriate fragment from the user selection. Comment before function (248-259) describes function action.
parsing one or more of a plurality of elements in the document type definition type selected;	Exhibit A, pages 9-10 shows the plurality of elements in a DTD in the Franklin specification of fragment and servable DTDs. Further Exhibit A, pages 9-10 refer to the UI types, i.e., requirements for user input and

	Exhibit A, pages 13-14 show an example of a servable DTD. Exhibit A, page 25 "Editor UI Widgets" the DTD is parsed and based on the DATATYPE generates the UI widget. Here we see the mapping from DATATYPE to java widget (e.g., string => JtextField) Exhibit A, pages 13-14 shows an example of a servable DTD and the set of elements that makes up that servable DTD. The system parses this plurality of elements to create the user interface; an example of such a user interface is shown on the right panel of Exhibit I, page 2 under <IMAGEFRAGEMENT 3: title> Exhibit L, lines 114-134) "/** * For the given DTD and content model node, create appropriate input widgets and add to the JPanel. ...." this comment indicates the function **createInterfaceForModel** creates the widgets based on the DTD definition and the elements of the content. Describes use of DATATYPE to select the widget.

We, the undersigned, declare all of the above statements are made on our own knowledge, the above statements are true and correct, and the above statements are made on information that we believe to be true. We understand that false statements or concealment in obtaining a patent will subject us to fine and/or imprisonment or both (18 U.S.C. §1001) and may jeopardize the validity of the above identified patent application or any application issuing therefrom.

Louis WEITZMAN	Sara ELO DEAN	Dikran S. MELIKSETIAN
October 4, 2006	October ___, 2006	October ___, 2006

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Appl. No.	:	09/748,716	Confirmation No. 5358
Applicant	:	Sara Elo DEAN et al.	
Filed	:	December 22, 2000	
TC/A.U.	:	2173	
Examiner	:	Brian J. DETWILER	
Docket No.	:	POUG920000205US1	
Customer No.	:	23334	

37 C.F.R. 1.131 DECLARATION

I, each and every one of the undersigned inventors of the above-referenced patent application, hereby declare the following:

1) Claims 1-9, 11-31, and 33-39 in our above-identified patent application were rejected under 35 U.S.C. §102(e) and claims 10 and 32 were rejected under 35 U.S.C. § 103(a) based on U.S. Patent Publication No. 2002/0085020 A1 to Carroll, Jr., entitled "XML-Based Graphical User Interface Application Development Toolkit" filed on September 14, 2001, with a priority date of September 14, 2000 ("Carroll").

2) The invention described in the above-referenced patent application was reduced to a writing prior to the September 14, 2000 priority date of Carroll. In particular, *Franklin Content Management Prototype* documentation (exhibit A), upon which the above referenced patent application was based, is attached herewith. The documentation is a comprehensive specification and installation of the inventive system (see the table of contents of this document for the full detail) created and used by the inventors prior to the September 14, 2000 priority date of Carroll and demonstrating features of the presently claimed invention. It includes everything from an installation guide, configuration, setup of the DB and a Franklin workspace for content management, setting up of users, roles, and includes code snippets of communication between components and error codes.

3) Additionally, the invention described in the above-referenced patent application was reduced to actual practice prior to the September 14, 2000 priority date of Carroll. Proof of actual reduction to practice upon which the presently claimed invention was based is attached herewith and will be described in detail below.

4) Submitted herewith as evidence of actual reduction to practice prior to the September 14, 2000 priority date of Carroll are the following exhibits:

Exhibit A) In particular, *Franklin Content Management Prototype* documentation (exhibit A), upon which the above referenced patent application was based, is attached herewith. (PDF File Pages 1-50)

Exhibit B) Assignments passed out to users prior to the September 14, 2000

priority date of Carroll to test users who were evaluating the integration between two systems: the present invention and "Kittyhawk" prior to the September 14, 2000 priority date of Carroll. The scenarios ask users to do different actions in the present invention's UI, which would show that there was a running system that could support users prior to the September 14, 2000 priority date of Carroll. The document describes the integration of the two systems, and shows the request/responses part of the communication between the two systems. (PDF File Pages 51-70)

Exhibit C) A copy of a State chart of the invention's DB with each possible state of a fragment when stored in the invention's DB. The State chart was created and used by the inventors prior to the September 14, 2000 priority date of Carroll and demonstrates features of the presently claimed invention. (PDF File Pages 70-71)

Exhibit D) Copies of HTML pages created by the inventors prior to the September 14, 2000 priority date of Carroll and demonstrating features of the presently claimed invention. The HTML pages describe to users how to install the inventive client and issue commands to manage documents, such as Check in, Check out, review, publish and describes the fragment/servable relationship to users. (PDF File Pages 72-75)

Exhibit E) A synthesis of all feedback from a user acceptance testing of the Invention, run prior to the September 14, 2000 priority date of Carroll. It includes a list of things users liked and did not like, which evidences that users were using the running end-to-end inventive system with features of the presently claimed invention prior to the September 14, 2000 priority date of Carroll. (PDF File Pages 76-90)

Exhibit F) A copy of brief notes identified during a code review of the Invention's server code made prior to the September 14, 2000 priority date of Carroll. (PDF File Page 91)

Exhibit G) An email correspondence to persons other than the inventors of the present invention, listing the internet address for accessing, and instructions on how to use, the working prototype system created and used by the inventors prior to September 14, 2000 priority date of Carroll and demonstrating features of the presently claimed invention. (PDF File Pages 92-94)

Exhibit H) An email correspondence with reviewer feedback on the working prototype system created and used by the inventors prior to the September 14, 2000 priority date of Carroll and demonstrating features of the presently claimed invention. (PDF File Pages 95-98)

Exhibit I) Copies of several screenshots of the working prototype system created and used by the inventors prior to September 14, 2000 priority date of Carroll and demonstrating features of the presently claimed invention. These screenshots show lists of XML documents having content objects and content fragments which are

named and linked through the entry fields. (PDF File Pages 99-101)

Exhibit J) A copy of a section of the source code file that was created and used by the inventors prior to September 14, 2000 priority date of Carroll and that implemented part of a working prototype system that performed features of the presently claimed invention. (PDF File Pages 102-103)

Exhibit K) A copy of the source code FranklinEditor.java that was created and used by the inventors prior to September 14, 2000 priority date of Carroll and that implemented part of a working prototype system that performed features of the presently claimed invention. (PDF File Pages 1-31)

Exhibit L) A copy of the source code InterfaceMaker.java that was created and used by the inventors prior to September 14, 2000 priority date of Carroll and that implemented part of a working prototype system that performed features of the presently claimed invention. (PDF File Pages 1-16)

5) The evidence submitted herewith supports the reduction to practice. The following table is submitted to show how each claim element is supported and that the test results unequivocally establish this software existed and worked for its intended purpose.

Claim1 is an example. The other independent claims (18, 23, & 39) recite identical limitations.

Claim 1: A method on an information processing unit for performing steps for assembling, with a user interface (UI), a document that conforms to a particular document type definition, the method comprising:

receiving a user selection for a document type	Exhibit B, page 3 under Scenario I, step 2.3, describes the step of creating the appropriate fragment "*Start the task to create the appropriate fragment, fill it in, and check it in.*" Exhibit A, page 25 "Create new content" paragraph shows the process of creating new content. Here the user must select a document type from the **File > New Fragment** menu or the **File > New Page** menu. These menus list all the available DTD types and it generates a template, for display in the UI, from the DTD selected. Exhibit A, page 25 "Create new content" and

	Exhibit D, page 2 "Create New Fragment" and "Create New Page" sections describe the procedure that a user has to perform to select and create a new document abiding to a document type. The system point of view of this process is the "reception of a user selection for a document type". Exhibit K, line 1095 Function **getToolBarPane** shows creation of UI to allow user to select a "new fragment or page" Lines 1099-1105 creates the button to do the action. e.g., iv_newButton.setToolTipText ("Create new fragment or page"); Exhibit K at line 760 Function **getFragmentTypeMenu** produces the menu for the user to select the type of DTD. Comment at 752-758 describes the function
selecting one of a plurality of document type definition types based upon the document type received;	Exhibit A, page 25, "Create new content" Here we are selecting a DTD based on the user selection of a document type. When the user has selected a type of document to create from the menu, the system retrieves the correct DTD from the "**appropriate URL**" and then generates the UI (template). Each URL represents a different DTD to use. Exhibit A, page 25 "Create new content" and Exhibit D, page 2 "Create New Fragment" and "Create New Page" sections. The user is presented with a list of types and selects one. The system selects the document type definition (DTD) corresponding to the chosen document type. Exhibit K at line 260-270 **createFragment**(String lv_name) creates the appropriate fragment from the user selection. Comment before function (248-259) describes function action.
parsing one or more of a plurality of elements in the document type definition type selected;	Exhibit A, pages 9-10 shows the plurality of elements in a DTD in the Franklin specification of fragment and servable DTDs. Further Exhibit A, pages 9-10 refer to the UI types, i.e., requirements for user input and

	Exhibit A, pages 13-14 show an example of a servable DTD. Exhibit A, page 25 "Editor UI Widgets" the DTD is parsed and based on the DATATYPE generates the UI widget. Here we see the mapping from DATATYPE to java widget (e.g., string => JtextField) Exhibit A, pages 13-14 shows an example of a servable DTD and the set of elements that makes up that servable DTD. The system parses this plurality of elements to create the user interface; an example of such a user interface is shown on the right panel of Exhibit I, page 2 under <IMAGEFRAGEMENT 3: title> Exhibit L, lines 114-134) "/** * For the given DTD and content model node, create appropriate input widgets and add to the JPanel. ...." this comment indicates the function **createInterfaceForModel** creates the widgets based on the DTD definition and the elements of the content. Describes use of DATATYPE to select the widget.

We, the undersigned, declare all of the above statements are made on our own knowledge, the above statements are true and correct, and the above statements are made on information that we believe to be true. We understand that false statements or concealment in obtaining a patent will subject us to fine and/or imprisonment or both (18 U.S.C. §1001) and may jeopardize the validity of the above identified patent application or any application issuing therefrom.

Louis WEITZMAN	Sara ELO DEAN	Dikran S. MELIKSETIAN
October ___, 2006	October 9, 2006	October ___, 2006

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Appl. No.	:	09/748,716	Confirmation No. 5358
Applicant	:	Sara Elo DEAN et al.	
Filed	:	December 22, 2000	
TC/A.U.	:	2173	
Examiner	:	Brian J. DETWILER	
Docket No.	:	POUG920000205US1	
Customer No.	:	23334	

37 C.F.R. 1.131 DECLARATION

I, each and every one of the undersigned inventors of the above-referenced patent application, hereby declare the following:

1) Claims 1-9, 11-31, and 33-39 in our above-identified patent application were rejected under 35 U.S.C. §102(e) and claims 10 and 32 were rejected under 35 U.S.C. § 103(a) based on U.S. Patent Publication No. 2002/0085020 A1 to Carroll, Jr., entitled "XML-Based Graphical User Interface Application Development Toolkit" filed on September 14, 2001, with a priority date of September 14, 2000 ("Carroll").

2) The invention described in the above-referenced patent application was reduced to a writing prior to the September 14, 2000 priority date of Carroll. In particular, *Franklin Content Management Prototype* documentation (exhibit A), upon which the above referenced patent application was based, is attached herewith. The documentation is a comprehensive specification and installation of the inventive system (see the table of contents of this document for the full detail) created and used by the inventors prior to the September 14, 2000 priority date of Carroll and demonstrating features of the presently claimed invention. It includes everything from an Installation guide, configuration, setup of the DB and a Franklin workspace for content management, setting up of users, roles, and includes code snippets of communication between components and error codes.

3) Additionally, the invention described in the above-referenced patent application was reduced to actual practice prior to the September 14, 2000 priority date of Carroll. Proof of actual reduction to practice upon which the presently claimed invention was based is attached herewith and will be described in detail below.

4) Submitted herewith as evidence of actual reduction to practice prior to the September 14, 2000 priority date of Carroll are the following exhibits:

Exhibit A) In particular, *Franklin Content Management Prototype* documentation (exhibit A), upon which the above referenced patent application was based, is attached herewith. (PDF File Pages 1-50)

Exhibit B) Assignments passed out to users prior to the September 14, 2000

priority date of Carroll to test users who were evaluating the integration between two systems: the present invention and "Kittyhawk" prior to the September 14, 2000 priority date of Carroll. The scenarios ask users to do different actions in the present invention's UI, which would show that there was a running system that could support users prior to the September 14, 2000 priority date of Carroll. The document describes the integration of the two systems, and shows the request/responses part of the communication between the two systems. (PDF File Pages 51-70)

Exhibit C) A copy of a State chart of the invention's DB with each possible state of a fragment when stored in the invention's DB. The State chart was created and used by the inventors prior to the September 14, 2000 priority date of Carroll and demonstrates features of the presently claimed invention. (PDF File Pages 70-71)

Exhibit D) Copies of HTML pages created by the inventors prior to the September 14, 2000 priority date of Carroll and demonstrating features of the presently claimed invention. The HTML pages describe to users how to install the inventive client and issue commands to manage documents, such as Check in, Check out, review, publish and describes the fragment/servable relationship to users. (PDF File Pages 72-75)

Exhibit E) A synthesis of all feedback from a user acceptance testing of the invention, run prior to the September 14, 2000 priority date of Carroll. It includes a list of things users liked and did not like, which evidences that users were using the running end-to-end inventive system with features of the presently claimed invention prior to the September 14, 2000 priority date of Carroll. (PDF File Pages 76-90)

Exhibit F) A copy of brief notes identified during a code review of the invention's server code made prior to the September 14, 2000 priority date of Carroll. (PDF File Page 91)

Exhibit G) An email correspondence to persons other than the inventors of the present invention, listing the internet address for accessing, and instructions on how to use, the working prototype system created and used by the inventors prior to September 14, 2000 priority date of Carroll and demonstrating features of the presently claimed invention. (PDF File Pages 92-94)

Exhibit H) An email correspondence with reviewer feedback on the working prototype system created and used by the inventors prior to the September 14, 2000 priority date of Carroll and demonstrating features of the presently claimed invention. (PDF File Pages 95-98)

Exhibit I) Copies of several screenshots of the working prototype system created and used by the inventors prior to September 14, 2000 priority date of Carroll and demonstrating features of the presently claimed invention. These screenshots show lists of XML documents having content objects and content fragments which are

named and linked through the entry fields. (PDF File Pages 99-101)

Exhibit J) A copy of a section of the source code file that was created and used by the inventors prior to September 14, 2000 priority date of Carroll and that implemented part of a working prototype system that performed features of the presently claimed invention. (PDF File Pages 102-103)

Exhibit K) A copy of the source code FranklinEditor.java that was created and used by the inventors prior to September 14, 2000 priority date of Carroll and that implemented part of a working prototype system that performed features of the presently claimed invention. (PDF File Pages 1-31)

Exhibit L) A copy of the source code InterfaceMaker.java that was created and used by the inventors prior to September 14, 2000 priority date of Carroll and that implemented part of a working prototype system that performed features of the presently claimed invention. (PDF File Pages 1-16)

5) The evidence submitted herewith supports the reduction to practice. The following table is submitted to show how each claim element is supported and that the test results unequivocally establish this software existed and worked for its intended purpose.

Claim1 is an example. The other independent claims (18, 23, & 39) recite identical limitations.

Claim 1: A method on an information processing unit for performing steps for assembling, with a user interface (UI), a document that conforms to a particular document type definition, the method comprising:

receiving a user selection for a document type	Exhibit B, page 3 under Scenario I, step 2.3, describes the step of creating the appropriate fragment "*Start the task to create the appropriate fragment, fill it in, and check it in.*" Exhibit A, page 25 "Create new content" paragraph shows the process of creating new content. Here the user must select a document type from the **File > New Fragment** menu or the **File > New Page** menu. These menus list all the available DTD types and it generates a template, for display in the UI, from the DTD selected. Exhibit A, page 25 "Create new content" and

	Exhibit D, page 2 "Create New Fragment" and "Create New Page" sections describe the procedure that a user has to perform to select and create a new document abiding to a document type. The system point of view of this process is the "reception of a user selection for a document type". Exhibit K, line 1095 Function **getToolBarPane** shows creation of UI to allow user to select a "new fragment or page" Lines 1099-1105 creates the button to do the action. e.g., iv_newButton.setToolTipText ("Create new fragment or page"); Exhibit K at line 760 Function **getFragmentTypeMenu** produces the menu for the user to select the type of DTD. Comment at 752-758 describes the function
selecting one of a plurality of document type definition types based upon the document type received;	Exhibit A, page 25, "Create new content" Here we are selecting a DTD based on the user selection of a document type. When the user has selected a type of document to create from the menu, the system retrieves the correct DTD from the "**appropriate URL**" and then generates the UI (template). Each URL represents a different DTD to use. Exhibit A, page 25 "Create new content" and Exhibit D, page 2 "Create New Fragment" and "Create New Page" sections. The user is presented with a list of types and selects one. The system selects the document type definition (DTD) corresponding to the chosen document type. Exhibit K at line 260-270 **createFragment**(String lv_name) creates the appropriate fragment from the user selection. Comment before function (248-259) describes function action.
parsing one or more of a plurality of elements in the document type definition type selected;	Exhibit A, pages 9-10 shows the plurality of elements in a DTD in the Franklin specification of fragment and servable DTDs. Further Exhibit A, pages 9-10 refer to the UI types, i.e., requirements for user input and

	Exhibit A, pages 13-14 show an example of a servable DTD. Exhibit A, page 25 "Editor UI Widgets" the DTD is parsed and based on the DATATYPE generates the UI widget. Here we see the mapping from DATATYPE to java widget (e.g., string => JtextField) Exhibit A, pages 13-14 shows an example of a servable DTD and the set of elements that makes up that servable DTD. The system parses this plurality of elements to create the user interface; an example of such a user interface is shown on the right panel of Exhibit I, page 2 under <IMAGEFRAGEMENT 3: title> Exhibit L, lines 114-134) "/** * For the given DTD and content model node, create appropriate input widgets and add to the JPanel. ...." this comment indicates the function **createInterfaceForModel** creates the widgets based on the DTD definition and the elements of the content. Describes use of DATATYPE to select the widget.

We, the undersigned, declare all of the above statements are made on our own knowledge, the above statements are true and correct, and the above statements are made on information that we believe to be true. We understand that false statements or concealment in obtaining a patent will subject us to fine and/or imprisonment or both (18 U.S.C. §1001) and may jeopardize the validity of the above identified patent application or any application issuing therefrom.

Louis WEITZMAN	Sara ELO DEAN	Dikran S. MELIKSETIAN
October ___, 2006	October ___, 2006	October 04 2006

Franklin Content Management Prototype

Documentation

Draft May 2000

IBM Confidential

IBM Advanced Internet Technology Group (WebAhead)
For more information, contact
Sara Elo (saraelo@us.ibm.com) or
Dikran Meliksetian (meliksd1@us.ibm.com)

Franklin team members.
Peter Davis
Sara Elo
Abel Henry
Dikran Meliksetian
Jeff Milton
Louis Weitzman
Jessica Wu
Joe Zhou

Table of Contents

Overview

Content on the Next Generation internet needs to be highly adaptive. New interfaces and devices are emerging, the diversity of users is increasing, machines are acting more and more on users' behalf, and net activities are possible for a wide range of business, leisure, education, and research activities.

To achieve maximum flexibility and reuse, content needs to be broken down into richly tagged fragments that can be combined and rendered appropriately for the user, task, and context. The Franklin content management prototype builds on this premise. It provides an end-to-end process from content creation and meta-tagging to quality assurance and publishing.

Franklin integrates several IBM technologies for its five components: content store, meta-data store, dispatcher, services and user interfaces. A high-level view of the components is shown in Figure 1: Franklin Components.

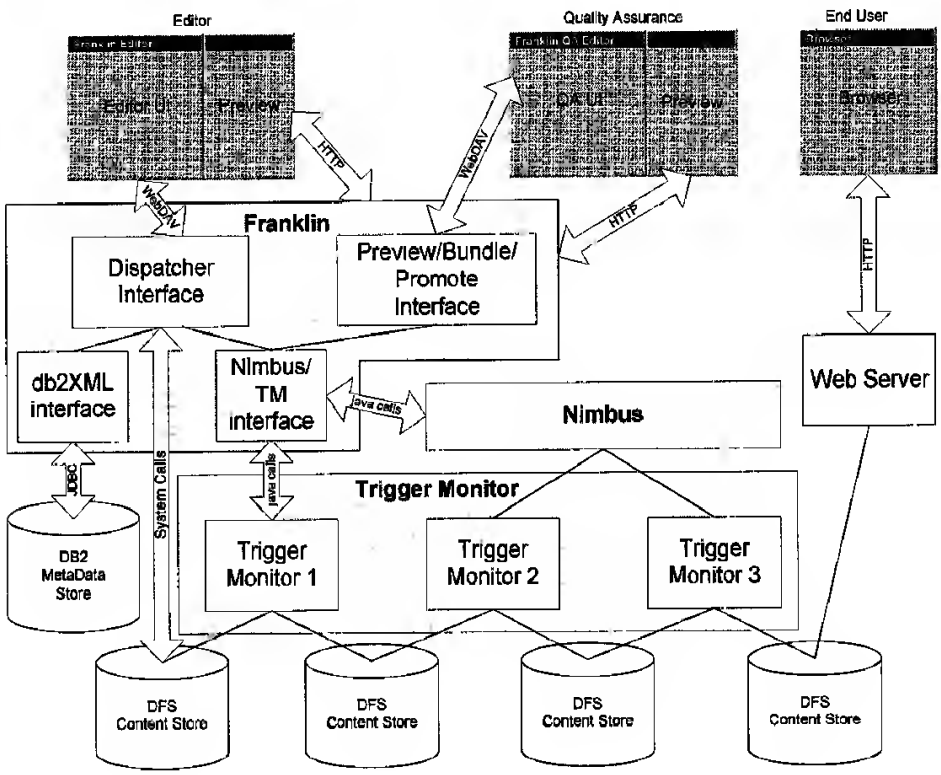


Figure 1: Franklin Components

The content store builds upon the Daedalus (a.k.a Trigger Monitor) technology from IBM Watson Research. [For full specification, see http://w3.watson.ibm.com/~challngr/papers/daedalus/index.html] Daedalus is designed to manage high numbers of rapidly changing content fragments. By maintaining an Object Dependency Graph, and by detecting changes to content, it manages pages on a web server or cached in a network router in a timely manner.

The meta-data store manages tags that describe the functional and semantic role of each content fragment within the information collection. They may describe what the content is about, who the target audience is, and its relationship to a taxonomy or other fragments. The meta-data store also supports efficient searches.

Networked services support the editor in content creation. They may assist the editor in meta-data creation, classification, summarization or translation. Instead of doing the task from beginning to end, the editor can accept, reject or modify the suggestions created by a service.

The dispatcher's task is to delegate incoming requests to the content store, meta-data store and the services. The dispatcher presents a consistent application programming interface to the user interfaces. This Franklin API abides to the Web protocol for Distributed Authoring and Versioning (WebDAV) and to the Distributed Authoring Search Language (DASL) specification.

The user interfaces communicate with the Franklin system through the API. Using the Editor UI, an editor can create and edit XML content fragments, upload XSL style sheets and multimedia objects, compose pages out of fragments, preview pages, review final published pages, and reject them or promote them to the final stage in the publishing flow.
The Franklin system has also been integrated with KittyHawk, an IBM Notes based workflow engine. This workflow module can be turned on or off depending on the application needs.

This document describes in detail the system requirements and setup, the architecture, components and features of Franklin. It covers the lessons learned, and provides a working example of a content collection managed with Franklin. In addition, it describes a lightweight version of Franklin, code-named Franklin Light, which satisfies the needs of small sites with no need for multiple Quality Assurance steps, or a scalable DB2 based search.

System Setup & Configuration

Before running an instance of Franklin to manage the content for a web site, you need to complete a number of installation steps. You also need to define the DTDs, the XSL style sheets, and the site map of the web site you intend to manage. This section outlines the required steps.

Step 1: Install Franklin Server

The Franklin Server runs on an AIX or NT server and requires the following software installed on the same machine:

- Apache Web Server v.1.3.6 or higher
- WebSphere Application Server v.2.0 or higher
- Java run-time environment 1.1.8

The Franklin server is distributed as a jar file, i.e., Franklin.jar. The distribution directory contains the following jar files that are required by Franklin: xml4j.jar, patbin132.jar, daedalus.jar, lotusxsl.jar, xerces.jar.

Deleted:

Download the Franklin Server and associated components from http://franklin.adtech.internet.ibm.com/franklin/downloads/index.html and place them in a directory accessible by the WebSphere Application Server.

Step 2: Install DB2 for Meta-Data Store

The DB2 database used by the Meta-data Store can run on the same machine or a different machine. It requires the following software:

1) DB2 6.1 with DB2 XML Extender 7.1
   Download DB2 XML Extenders 7.1 from IBM software website at http://www.software.ibm.com/db2. Currently, XML Extenders is supported on Windows NT, AIX and Solaris. If you decide to use the XML Extender Administration Wizard make sure you review the XML Extender Administration Wizard Readme file to ensure you have the software prerequisites, JDK 1.1.x or JRE v1.1.x and JFC 1.1 with Swing 1.1 or later.
2) JDBC for DB2 JDBC 1.20
   JDBC is included in the DB2 installation (db2java.zip) in the directory of sqllib/java.

The steps to install DB2 and enable DB2 XML Extenders (which require root authority on AIX):

1) Install a version of UDB higher than 5.2. We have tested DB2 XML on NT for UDB 5.2 and 6.1, and on AIX for UDB 6.1 for DB2 XML XColumn function.
2) Create a DB2 instance. In the included examples, we use the db2 instance name db2frnkl.
3) Install DB2 XML Extenders
4) Create a database in the instance. Also, create the tables and indexes based on the sample scripts we have provided.
5) Enable the database with XML Extenders
6) Start JDBC on a port. For example, "db2jstrt 4000" opens port 4000 for JDBC connections.

Step 3: Customize Server Initialization Files

Edit *franklinServletInitialization.properties* file and set the following variable to the desired directory in your setup:

baseDir – base directory for all Franklin related files.

Comment [LW1]: Should we name the properties files differently? E.g. franklinClient.properties vs. franklinServer.properties?

All other variables in *franklinServletInitialization.properties* are relative to *baseDir* and should not be changed:

dtdDir - directory for DTD and entity files
xmlDir - root of the directory hierarchy for XML files

assetsDir - directory for all directories browsable by client UI, i.e. *xslDir, publishDir, multimediaDir*
xslDir - directory for XSL style sheets
publishDir - root of the directory hierarchy for HTML, HDML or DHTML files
multimediaDir - root of the directory hierarchy for images, graphics, video and audio files

Edit *metastore.ini* file and set the following variables to the desired directory in your setup:

MetaStoreServerIP - database host machine name
MetaStoreServerPort - JDBC port number
MetaStoreServerDBName – database name
MetaStoreServerUserID - database user name
MetaStoreServerPassword - database password for the above user
MetaStoreServerDriverClassName - database JDBC driver name
MetaStoreServerInitialConnection - number of initial connections to the database
MetaStoreServerIncrement - number of additional connections to database
MetaStoreCheckInXMLDir - temporary directory for XML files checked into meta store
MetaStoreDADDir – directory for DAD files
MetaStoreCacheSearchDir - directory for cached XML Search results

Step 4: Configure WebSphere Application Server

Start the WebSphere Administrative Console, refer to the WebSphere Quick Beginnings guide for details

1. In the Tasks tab of the console, select Configure a Web Application and click the start task button
   a. Specify the Web Application Name, e.g., FranklinServer, click Next.
   b. Choose the servlet engine, e.g., the ServletEngine in the Default Server of the Default Host, click Next
   c. Specify the Web Application Web Path, e.g., /franklinserver, click Next
   d. Specify the CLASSPATH:
      i. Add each of the jar files in the Franklin distribution to the classpath, i.e., franklin.jar, daedalus.jar, xml4j.jar, lotusxsl.jar, xerces.jar, patbin132.zip
      ii. Add the db2java.zip file to the classpath, the db2java.zip file is distributed with DB2, it is found under the sqllib/java subdirectory of the database instance home directory
      iii. Click Finished
2. In the same Tasks tab, select Add a Servlet and click the start task button
   a. Select Yes to "Do you want to select an existing Servlet jar file or Directory that contains Servlet classes" and click Next
   b. Specify the path of the directory where franklin.jar is located, click Next
   c. Select the Web Application that was created in the previous step, click Next
   d. Select the Create User-Defined Servlet option, click Next
   e. Specify the Servlet Name, e.g., dispatcher
   f. Specify the Servlet Class Name as com.ibm.adtech.franklin.server.dispatcher.Dispatcher

g. Specify the Servlet Web Path List, for example /franklinserver/dispatcher, this is the web path that should be used by the client to access the franklin server, click next
h. Add an init parameter with Init Parm Name as baseDir and Init Parm Value equal to the directory where the franklin server configuration files are stored, e.g., /franklin/data/config
i. Select the True option for Load at Startup; click Finished

3. The configuration is complete you need to start the service, select the Topology tab
   a. Prior to starting the application, make sure that the database instance is running and that jdbc daemon is active (see previous section)
   b. Expand the topology tree and select the newly created application. The application will appear under the servlet engine that was selected in step 1.b
   c. Right click the selection and on the popup menu select Restart Application
4. The Franklin Server should be available at this point. In order to verify that everything is in order view the log files of WebSphere

Step 5: Install Franklin Client

The Franklin Client Java Application has been tested on Windows98/2000/NT.

Download the Franklin Client Application Installer FranklinEditor.exe from http://franklin.adtech.internet.ibm.com/franklin/downloads/index.html and run it. In addition to the Franklin Client, the following Java packages are required and are automatically installed by the Installer:

- Java 1.1.8 run-time environment with Swing JFC1.1.1
- XML4J package
- WebDav package

The Franklin Client Application Installer also creates the subdirectories required by the client under the chosen installation directory. You can change these directories as described in the next paragraph.

After installation, customize the initialization file *franklin.properties* located in the root directory where you installed the Franklin Client application. You need to edit the variable *browserPath* to define the location of the web browser you wish to use to preview pages. Also, you can edit the variable *tempDir* if you wish to change the directory where temporary files are stored.

Comment [LW2]: Should variable names in the client properties file be more similar to the names of the variables in the dispatcher's properties file

dispatcher = *http://adtech.ibmus2.ibm.com/franklinservlet/*
initXMLFile = *xml/franklin_init.xml*
modify browserPath to point to the web browser you wish to use for preview
browserPath = *c:/Program Files/Internet Explorer/iexplore.exe*
modify tempDir to point to the directory where temporary files will be stored
tempDir = *./tmp/*
tempMediaDir = *media/*
tempHTMLDir = *html/*
tempXSLDir = *xsl/*
standaloneP = *false*
validateP = *true*

Step 6: Define Document Type Definitions (DTD)

Franklin manages two types of content objects, *fragments* and *servables*.

A fragment is a content object that can be reused on several pages:

- a *simple fragment* is a self contained XML file containing text data and metadata – for example, a product specification
- a *compound fragment* is an XML file that contains metadata and points to an accompanying file such as a video or image file, an XSL style sheet, or a hand-crafted HTML page
- an *index fragment* is an automatically updated XML file that indexes any number of servables - for example a panel listing the five latest press release [**Future**: index fragments not available in current implementation]

A servable is an XML file that contains the text and meta-data for one final published page and imports reusable content from one or more fragments, and points to one of more style sheet fragments.

Figure 2 shows a product page servable which includes content from six fragments, namely three text fragments, one image fragment and two style sheet fragments, and results in two final published pages.

Insert Figure 2 here

Before beginning to manage a content collection, you need to define the document type definitions, or DTDs, for each class of fragment and servable that will be managed by the application. Franklin uses the syntax of DTDs to define a document type. [See the XML specification at http://www.w3.org/TR/REC-xml]

In order for Franklin to manage DTDs correctly, all DTDs must abide to the Franklin following specifications:

Franklin specification of fragment and servable DTDs

1. The root element, to which you can give a meaningful name, must have a child node called `SYSTEM` with the children nodes `FRAGMENTID, CREATOR, MODIFIER, CREATIONTIME, LASTMODIFIEDTIME, PAGETYPE and CONTENTSIZE.` The `NAME` attribute of `PAGETYPE` must be set to either `"FRAGMENT"` or `"SERVABLE"`.

```
<!ELEMENT ROOT          (SYSTEM, ...)>

<!ELEMENT SYSTEM        (FRAGMENTID,
                        CREATIONTIME,
                        LASTMODIFIEDTIME,
                        CREATOR,
                        MODIFIER,
                        PAGETYPE
                        CONTENTSIZE?)>
```

```
<!ELEMENT FRAGMENTID        (#PCDATA)>
<!ELEMENT CREATIONTIME      (#PCDATA)>
<!ELEMENT LASTMODIFIEDTIME  (#PCDATA)>
<!ELEMENT CREATOR           (#PCDATA)>
<!ELEMENT MODIFIER          (#PCDATA)>
<!ELEMENT CONTENTSIZE       (#PCDATA)>
<!ELEMENT PAGETYPE          (#PCDATA)>
<!ATTLIST PAGETYPE          NAME (FRAGMENT|SERVABLE) "FRAGMENT" #FIXED>
```

2. All items editable in the Editor UI need to be elements of the DTD, not attributes. For example,

```
<!ELEMENT TITLE             (#PCDATA)>
<!ELEMENT SHORTDESCRIPTION  (#PCDATA)>
<!ELEMENT CATEGORY          (#PCDATA)>
```

3. All elements to be indexed for search must be of type PCDATA, and must contain the attribute SEARCH set to YES. For example,

```
<!ELEMENT ROOT               (TITLE, SHORTDESCRIPTION, CATEGORY, ...)>
<!ELEMENT TITLE             (#PCDATA)>
<!ELEMENT SHORTDESCRIPTION  (#PCDATA)>
<!ELEMENT CATEGORY          (#PCDATA)>
<!ATTLIST TITLE             SEARCH (YES|NO) "YES" #FIXED>
<!ATTLIST SHORTDESCRIPTION  SEARCH (YES|NO) "YES" #FIXED>
<!ATTLIST CATEGORY          SEARCH (YES|NO) "YES" #FIXED>
```

Future: Need to add SEARCH attribute. The SEARCH attribute will allow Franklin to automatically generate the DAD mapping for the DB2 XML Extenders..

4. Include the external entity reference that defines the user interface widgets recognized by the Franklin Editor UI. Each element that needs to be editable in the Editor UI must be of type PCDATA and contain the DATATYPE attribute set to the appropriate UI type.

```
<!ENTITY  % UITYPES      SYSTEM
"http://franklinserver/franklin/dtd/uitypes.txt">

<!ATTLIST TITLE             DATATYPE (%UITYPES;)      "STRING"    #FIXED>
<!ATTLIST SHORTDESCRIPTION  DATATYPE (%UITYPES;)      "LONGTEXT"  #FIXED
                            PARSE    (TRUE)           "TRUE"      #FIXED>
```

If you wish a LONGTEXT widget to allow an editor to enter a limited set of HTML tags, add the PARSE attribute and set it to true. The supported HTML are:

```
<p>, <ul>, <ol>, <sl>, <dl>, <dt>, <dd>, <li>, <div>
```

The file *uitypes.txt* is fixed and provided in the Franklin install in the *dtdDir* in the *franklin.properties* file. It contains the list of all UI widgets known to the Editor UI. (See section Editor UI Widgets for a detailed description of UITYPES)

```
DATE | INTEGER | STRING | SHORTTEXT | LONGTEXT | CHOICE | BROWSESERVER |
BROWSELOCAL | ASSOCLIST
```

5. An element can appear as a drop-down menu in the Editor UI and restrict the editor to choose the value from a predefined set. To accomplish this, set the DATATYPE attribute to the UITYPE "CHOICE" and the CHOICES attribute to a default value from a list of options. The options can be defined as an external entity for reuse across many DTDs.

```
<!ENTITY  % CATEGORYDEFS      SYSTEM
"http://franklinserver/franklin/dtd/categorydefs.txt">
<!ATTLIST CATEGORY          DATATYPE (%UITYPES;)       "CHOICE"     #FIXED
                            CHOICES  (%CATEGORYDEFS;) "NONE"       #REQUIRED>
```

For example, the options for CATEGORY could be defined as the types of Netfinity servers:

```
NONE | Netfinity_8500R | Netfinity_7000_M10 | Netfinity_5500_M10 |
Netfinity_5600 | Netfinity_5500
```

The Editor UI assumes that if the first word in the set of CHOICES is the string NONE, and the editor selects it, the element will not appear in the XML document.

6. A fragment can include other fragments as subfragments. If so, the entity reference that defines all subfragment types must be included in the DTD. The declaration of a subfragment must contain the SUBFRAGMENTTYPE attribute set to the appropriate type.

```
<!ENTITY % SUBFRAGMENTTYPES SYSTEM
"http://franklinserver/franklin/dtd/subfragmenttypes.txt">
<!ELEMENT SUBFRAGMENT (#PCDATA)>
<!ATTLIST SUBFRAGMENT SUBFRAGMENTTYPE (%SUBFRAGMENTTYPES;) "IMAGEFRAGMENT"
#FIXED>
```

Future: the subfragment syntax will be replaced by the XLink syntax once it becomes a W3 recommendation and XML4J and LotusXSL support the syntax. Until then, we will use subfragment elements as way to include content from another fragment.

An example of a fragment DTD, listfragment.dtd:

```
<!ENTITY % SUBFRAGMENTTYPES SYSTEM
"http://franklinserver/franklin/dtd/subfragmenttypes.txt">
<!ENTITY  % CATEGORYDEFS      SYSTEM
"http://franklinserver/franklin/dtd/categorydefs.txt">
<!ENTITY  % UITYPES      SYSTEM
"http://franklinserver/franklin/dtd/uitypes.txt">

<!ELEMENT LISTFRAGMENT (SYSTEM, TITLE, SHORTDESCRIPTION?, CATEGORY*,
                        LISTITEM+)>
<!ELEMENT SYSTEM       (FRAGMENTID, CREATOR, MODIFIER, CREATIONTIME,
                        LASTMODIFIEDTIME, PAGETYPE, CONTENTSIZE?)>
<!ELEMENT FRAGMENTID        (#PCDATA)>
<!ELEMENT CREATIONTIME      (#PCDATA)>
<!ELEMENT LASTMODIFIEDTIME  (#PCDATA)>
<!ELEMENT CONTENTSIZE       (#PCDATA)>
<!ELEMENT CREATOR           (#PCDATA)>
<!ELEMENT MODIFIER          (#PCDATA)>
<!ELEMENT PAGETYPE         (#PCDATA)>
<!ELEMENT TITLE             (#PCDATA)>
<!ELEMENT SHORTDESCRIPTION (#PCDATA)>
```

```
<!ELEMENT CATEGORY          (#PCDATA)>
<!ELEMENT LISTITEM          (LISTTITLE?, DESCRIPTION?, LINK?, FOOTNOTE?)>
<!ELEMENT LISTTITLE         (#PCDATA)>
<!ELEMENT DESCRIPTION       (#PCDATA)>
<!ELEMENT LINK              (#PCOATA)>
<!ELEMENT FOOTNOTE          (#PCDATA)>

<!ATTLIST TITLE             DATATYPE (%UITYPES;)     #FIXED      "STRING"
                            SEARCH   (YES|NO) "YES" #FIXED>

<!ATTLIST SHORTDESCRIPTION DATATYPE (%UITYPES;)     #FIXED      "SHORTTEXT"
                            SEARCH   (YES|NO) "YES" #FIXED>
<!ATTLIST CATEGORY          DATATYPE (%UITYPES;)       #FIXED     "CHOICE"
                            CHOICES  (%CATEGORYDEFS;) #IMPLIED
                            SEARCH   (YES|NO) "YES" #FIXED>
<!ATTLIST LISTTITLE          DATATYPE (%UITYPES;)      #FIXED      "STRING">
<!ATTLIST DESCRIPTION       DATATYPE (%UITYPES;)       #FIXED      "STRING">
<!ATTLIST LINK              DATATYPE  (%UITYPES;)      #FIXED      "STRING">
<!ATTLIST FOOTNOTE          DATATYPE  (%UITYPES;)      #FIXED      "STRING">
```

Franklin specification of compound fragment DTDs

A compound fragment contains a pointer to an accompanying file, such as a multimedia file, an XSL style sheet or a hand-crafted HTML file (i.e. ones that are not generated from XML by Franklin) The accompanying file is encoded as a binary object into the fragment for the duration of the communication between Editor UI and Franklin Server. Before check-in to the server, the Editor UI encodes the file as a binary object into the XML fragment. At the receiving end, the dispatcher extracts it and decodes into the original format. The reverse happens at check-out. This allows a single communication between the client and server when exchanging this type of fragments.

1. A compound fragment DTD must use the following syntax to declare the inclusion of an external file:

```
<!ELEMENT CONTENT           (#PCDATA)>
<!ELEMENT CONTENTDIR        (#PCDATA)>
<!ELEMENT CONTENTFILENAME   (#PCDATA)>
<!ATTLIST CONTENT          DATATYPE (%UITYPES;)  #FIXED      "STRING">
<!ATTLIST CONTENTDIR       OATATYPE (%UITYPES;)  #FIXED      "BROWSESERVER">
<!ATTLIST CONTENTFILENAME DATATYPE (%UITYPES;)  #FIXED      "BROWSELOCAL">
```

Franklin specification of group index fragment DTDs

Future: To be filled in

Franklin specification of servable DTDs

In the Franklin system, servables always result in one of more final published pages. The DTD must indicate the names of the XSL style sheets it can use for layout and where to publish the resulting pages.

1. A servable DTD must contain the following declarations:

```
<!ELEMENT PUBLISHINFO       (STYLESHEET, PUBLISHDIR, PUBLISHFILENAME)>
<!ELEMENT STYLESHEET        (#PCDATA)>
<!ELEMENT PUBLISHDIR        (#PCDATA)>
<!ELEMENT PUBLISHFILENAME   (#PCDATA)>
<!ATTLIST STYLESHEET       DATATYPE (%UITYPES;) #FIXED "CHOICE"
          CHOICES (styleshheet1.xsl|stylesheet2.xsl) #IMPLIED>
<!ATTLIST PUBLISHDIR       DATATYPE  (%UITYPES;) #FIXED "BROWSESERVER">
<!ATTLIST PUBLISHFILENAME DATATYPE  (%UITYPES;) #FIXED "STRING">
```

2. A servable can include one or more subfragments. Each subfragment serves a specific role within the servable and can be named in a meaningful way, for example MAINPHOTO, HIGHLIGHTS etc. Each subfragment must have an attribute that indicates the type of subfragment to include. The syntax to include a subfragment in a servable follows:

```
<!ELEMENT MAINPHOTO         (#PCDATA)>
<!ELEMENT HIGHLIGHTS        (#PCDATA)>
<!ATTLIST MAINPHOTO       DATATYPE (%UITYPES;)  #FIXED "STRING"
          SUBFRAGMENTTYPE (%SUBFRAGMENTTYPES;)  #FIXED "IMAGEFRAGMENT">
<!ATTLIST HIGHLIGHTS      DATATYPE (%UITYPES;)  #FIXED "STRING"
          SUBFRAGMENTTYPE (%SUBFRAGMENTTYPES;)  #FIXED "LISTFRAGMENT">
```

3. A servable can be included in an automatically generated group index fragment.

To be filled in

An example of a servable DTD, productpage.dtd:

```
<!ENTITY % UNIVERSAL        SYSTEM
"http://franklinserver/franklin/dtd/universal.dtd">
<!ENTITY % SUBFRAGMENTTYPES SYSTEM
"http://franklinserver/franklin/dtd/subfragmenttypes.txt">
<!ENTITY  % CATEGORYDEFS     SYSTEM
"http://franklinserver/franklin/dtd/categorydefs.txt">
<!ELEMENT PRODUCTPAGE (SYSTEM, TITLE, SOURCE?, COMMENT?, SHORTDESCRIPTION?,
                       LONGDESCRIPTION?, KEYWORD*, CATEGORY*, RELATEDLINK*,
                       PUBLISHINFO+, BRANDNAVIGATION, MAINPHOTO, GLANCE,
                       HIGHLIGHTS , GROUPINDEX*)>
<!ELEMENT SYSTEM      (FRAGMENTID, CREATOR, MODIFIER, CREATIONTIME,
                       LASTMODIFIEDTIME, PAGETYPE, CONTENTSIZE?)>
<!ELEMENT FRAGMENTID        (#PCDATA)>
<!ELEMENT CREATIONTIME      (#PCDATA)>
<!ELEMENT LASTMODIFIEDTIME  (#PCDATA)>
<!ELEMENT CONTENTSIZE       (#PCDATA)>
<!ELEMENT CREATOR           (#PCDATA)>
<!ELEMENT MODIFIER          (#PCDATA)>
<!ELEMENT PAGETYPE           (#PCDATA)>
<!ELEMENT TITLE             (#PCDATA)>
<!ELEMENT SOURCE            (#PCDATA)>
<!ELEMENT COMMENT           (#PCDATA)>
<!ELEMENT SHORTDESCRIPTION  (#PCDATA)>
<!ELEMENT LONGDESCRIPTION   (#PCDATA)>
<!ELEMENT KEYWORD           (#PCDATA)>
```

```
<!ELEMENT CATEGORY          (#PCDATA)>
<!ELEMENT RELATEDLINK       (URL, LINKTITLE)>
<!ELEMENT URL               (#PCDATA)>
<!ELEMENT LINKTITLE         (#PCDATA)>
<!ELEMENT DOCTYPE           (#PCDATA)>
<!ELEMENT DTDURL            (#PCDATA)>
<!ELEMENT PUBLISHINFO  (STYLESHEET, PUBLISHDIR, PUBLISHFILENAME)>
<!ELEMENT STYLESHEET        (#PCDATA)>
<!ELEMENT PUBLISHDIR        (#PCDATA)>
<!ELEMENT PUBLISHFILENAME   (#PCDATA)>
<!ELEMENT BRANDNAVIGATION   (#PCDATA)>
<!ELEMENT MAINPHOTO         (#PCDATA)>
<!ELEMENT GLANCE            (#PCDATA)>
<!ELEMENT HIGHLIGHTS        (#PCDATA)>
<!ELEMENT GROUPINDEX        (#PCDATA)>
<!ATTLIST TITLE             DATATYPE (%UITYPES;)       #FIXED      "STRING">
<!ATTLIST SOURCE            DATATYPE (%UITYPES;)       #FIXED      "STRING">
<!ATTLIST COMMENT           DATATYPE (%UITYPES;)     FIXED       "STRING">
<!ATTLIST SHORTDESCRIPTION DATATYPE (%UITYPES;)     #FIXED       "SHORTTEXT">
<!ATTLIST LONGDESCRIPTION  DATATYPE (%UITYPES;)       #FIXED
"SHORTTEXT">
<!ATTLIST KEYWORD           DATATYPE (%UITYPES;)       #FIXED      "STRING">
<!ATTLIST CATEGORY          DATATYPE (%UITYPES;)       #FIXED      "CHOICE"
                             CHOICES  (%CATEGORYDEFS;) #IMPLIED>
<!ATTLIST URL               DATATYPE (%UITYPES;)       #FIXED      "STRING">
<!ATTLIST LINKTITLE         DATATYPE (%UITYPES;)       #FIXED      "STRING">
<!ATTLIST BRANDNAVIGATION   DATATYPE (%UITYPES;)       #FIXED    "STRING"
          SUBFRAGMENTTYPE (%SUBFRAGMENTTYPES;)  #FIXED LISTFRAGMENT"
<!ATTLIST MAINPHOTO      DATATYPE  (%UITYPES;)         #FIXED    "STRING"
          SUBFRAGMENTTYPE (%SUBFRAGMENTTYPES;)  #FIXED "IMAGEFRAGMENT">
<!ATTLIST GLANCE         DATATYPE   (%UITYPES;)       #FIXED     "STRING"
          SUBFRAGMENTTYPE (%SUBFRAGMENTTYPES;)  #FIXED "LISTFRAGMENT">
<!ATTLIST HIGHLIGHTS     DATATYPE   (%UITYPES;)  #FIXED    "STRING"
          SUBFRAGMENTTYPE (%SUBFRAGMENTTYPES;)  #FIXED "LISTFRAGMENT">
<!ATTLIST STYLESHEET     DATATYPE   (%UITYPES;) #FIXED "CHOICE"
          CHOICES  (web_product_index.xsl | pda_product_index.xsl)
           #IMPLIED>
<!ATTLIST PUBLISHDIR     DATATYPE  (%UITYPES;)  #FIXED "BROWSESERVER">
<!ATTLIST PUBLISHFILENAME DATATYPE(%UITYPES;)  #FIXED "STRING">
<!ATTLIST GROUPINDEX      DATATYPE (%UITYPES;)  #FIXED "STRING"
          SUBFRAGMENTTYPE (%SUBFRAGMENTTYPES;)  #FIXED "GROUPINDEX">
```

An example of a servable, 2-tsrv.xml, which abides to productpage.dtd:

```
<?xml version="1.0"?>
<!DOCTYPE PRODUCTPAGE SYSTEM "http://franklinserver/dtd/productpage.dtd">
<PRODUCTPAGE>
   <SYSTEM>
      <FRAGMENTID>2-tsrv.xml</FRAGMENTID>
      <CREATOR>Joe Moe</CREATOR>
      <MODIFIER>Jane Mane</MODIFIER>
      <CREATIONTIME>384738740383</CREATIONTIME>
      <LASTMODIFIEDTIME>384738740383</LASTMODIFIEDTIME>
      <PAGETYPE>Servable</PAGETYPE>
   </SYSTEM>
   <TITLE>Netfinity 8500R</TITLE>
```

```
   <SOURCE>IBM PC Company</SOURCE>
   <SHORTDESCRIPTION>Mainframe features bring extraordinary performance and
reliability to a rack-optimized 8-way server</SHORTDESCRIPTION>
  <LONGDESCRIPTION>A great value in 8-way servers, the new Netfinity 8500R
maximizes uptime and provides superior manageability for compute-intensive
business intelligence, transaction processing and server consolidation
projects. </LONGDESCRIPTION>
   <KEYWORD>New</KEYWORD>
   <KEYWORD>Server</KEYWORD>
   <KEYWORD>Pentium</KEYWORD>
   <CATEGORY>Netfinity 8500R</CATEGORY>
   <CATEGORY>Small and Medium Business</CATEGORY>
   <RELATEDLINK>
      <URL>ftp://ftp.pc.ibm.com/pub/pccbbs/pc_servers/8500rf.pdf</URL>
      <LINKTITLE>White paper</LINKTITLE>
   </RELATEDLINK>
   <PUBLISHINFO>
         <PUBLISHDIR>/web/netfinity/</PUBLISHDIR>
         <PUBLISHFILENAME>index.html</PUBLISHFILENAME>
         <STYLESHEET>web_product_index.xsl</STYLESHEET>
    </PUBLISHINFO>
    <PUBLISHINFO>

         <PUBLISHDIR>/pda/netfinity/</PUBLISHDIR>
         <PUBLISHFILENAME>index.html</PUBLISHFILENAME>
         <STYLESHEET>pda_product_index.xsl</STYLESHEET>
    </PUBLISHINFO>
    <GROUPINDEX>444-ifrg.xml</GROUPINDEX>
    <MAINPHOTO SUBFRAGMENTTYPE="IMAGEFRAGMENT">
         222-bfrg.xml</MAINPHOTO>
    <HIGHLIGHTS SUBFRAGMENTTYPE="LISTFRAGMENT">
         444-tfrg.xml</HIGHLIGHTS>
</PRODUCTPAGE>
```

Once all DTDs for the collection have been defined, save them in the directory defined by the variable *dtdDir* in the *franklin.properties* file.

After updating, adding or deleting any DTDs, update the files *configDir/dtds.xml* and *dtdDir/subfragmenttypes.txt* to reflect the current DTDs. Also remember to define a DAD file to for each DTD. (**Future**: DAD explanation should be expanded)

Step 7: Define Style Sheets

For each servable DTD, you need to define one or more XSL style sheets that will be assembled with the servable XML and the XML of any subfragments into the final published pages. A style sheet is written using the XSL syntax to produce HTML, DHTML, HDML or other desired output. [See the XSL Transformations syntax at http://www.w3.org/TR/xslt]

Comment [LW3]: mention LotusXSL and have url to alphaworks site

Franklin specification of XSL style sheets

Because the servable includes content from subfragments, the style sheet must be written to work on the so-called *expanded* servable. Before page assembly, a servable is temporarily rewritten to include the content of all its subfragments. Because the XLink standard has not been finalized,

XSL style sheets cannot access content stored in subfragment files outside the servable. Franklin implements a temporary solution that mimics the XLink functionality by expanding the servable. This is demonstrated by the expanded product page 2-tsrv.xml:

```
<?xml version="1.0"?>
<!DOCTYPE PRODUCTPAGE SYSTEM
"http://yourfranklinserver/dtd/productpage.dtd">
<PRODUCTPAGE>
   <SYSTEM>
      <FRAGMENTID>2-tsrv.xml</FRAGMENTID>
      <CREATOR>Joe Moe</CREATOR>
      <MODIFIER>Jane Mane</MODIFIER>
      <CREATIONTIME>384738740383</CREATIONTIME>
      <LASTMODIFIEDTIME>384738740383</LASTMODIFIEDTIME>
      <PAGETYPE>Servable</PAGETYPE>
   </SYSTEM>
   <TITLE>Netfinity 8500R</TITLE>
   <SOURCE>IBM PC Company</SOURCE>
   <SHORTDESCRIPTION>Mainframe features bring extraordinary performance and
reliability to a rack-optimized 8-way server</SHORTDESCRIPTION>
  <LONGDESCRIPTION>A great value in 8-way servers, the new Netfinity 8500R
maximizes uptime and provides superior manageability for compute-intensive
business intelligence, transaction processing and server consolidation
projects. </LONGDESCRIPTION>
   <KEYWORD>New</KEYWORD>
   <KEYWORD>Server</KEYWORD>
   <KEYWORD>Pentium</KEYWORD>
   <CATEGORY>Netfinity 8500R</CATEGORY>
   <CATEGORY>Small and Medium Business</CATEGORY>
   <RELATEDLINK>
      <URL>ftp://ftp.pc.ibm.com/pub/pccbbs/pc_servers/8500rf.pdf</URL>
      <LINKTITLE>White paper</LINKTITLE>
   </RELATEDLINK>
    <PUBLISHINFO>
          <PUBLISHDIR>/web/netfinity/</PUBLISHDIR>
          <PUBLISHFILENAME>index.html</PUBLISHFILENAME>
          <STYLESHEET>web_product_index.xsl</STYLESHEET>
    </PUBLISHINFO>
    <PUBLISHINFO>
          <PUBLISHDIR>/pda/netfinity/</PUBLISHDIR>
          <PUBLISHFILENAME>index.html</PUBLISHFILENAME>
          <STYLESHEET>pda_product_index.xsl</STYLESHEET>
    </PUBLISHINFO>
    <GROUPINDEX>444-ifrg.xml</GROUPINDEX>
    <MAINPHOTO SUBFRAGMENTTYPE="IMAGEFRAGMENT">
     <IMAGEFRAGMENT>
          <SYSTEM>
                <FRAGMENTID>222-bfrg.xml</FRAGMENTID>
                <CREATOR>BOB</CREATOR>
                <MODIFIER>BOB</MODIFIER>
                <CREATIONTIME>38473B7403B3</CREATIONTIME>
                <LASTMODIFIEDTIME>384738740383</LASTMODIFIEDTIME>
                 <PAGETYPE>Fragment</PAGETYPE>
                <CONTENTSIZE>400</CONTENTSIZE>
          </SYSTEM>
          <TITLE>The Netfinity 8500R large jpeg</TITLE>
          <SHORTDESCRIPTION>Netfinity 8500R</SHORTDESCRIPTION>
```

```
          <CONTENTDIR>/images/prod_images/</CONTENTDIR>
          <CONTENTFILENAME>8500R_large.jpg</CONTENTFILENAME>
          <CONTENT/>
       </IMAGEFRAGMENT>
     </MAINPHOTO>
     <HIGHLIGHTS SUBFRAGMENTTYPE="LISTFRAGMENT">
        <LISTFRAGMENT>
          <SYSTEM>
                <FRAGMENTID>444-tfrg.xml</FRAGMENTID>
                <CREATOR>BOB</CREATOR>
                <MODIFIER>BOB</MODIFIER>
                <CREATIONTIME>384738740383</CREATIONTIME>
                <PAGETYPE>Fragment</PAGETYPE>
                <LASTMODIFIEDTIME>384738740383</LASTMODIFIEDTIME>
          </SYSTEM>
          <TITLE>Highlights of the 8500R</TITLE>
          <CATEGORY>Netfinity 8500R</CATEGORY>
          <LISTITEM>
                      <TITLE>Light-Path Diagnostics</TITLE>
                      <BODY>Lighted guidance system to assist in quick
identification of failing components, similar to the lights in a copier
that identify the location of a paper jam.</BODY>
          </LISTITEM>
          <LISTITEM>
                      <TITLE>Active PCI technology</TITLE>
                      <BODY>Enables IBM's unique hot-add PCI, letting you
add client systems, balance network traffic or increase storage capacity
without shutting the system down. </BODY>
          </LISTITEM>
        </LISTFRAGMENT>
     </HIGHLIGHTS>
</PRODUCTPAGE>
```

Any Xpath expressions in the style sheet that refer to subfragment content will be local to the servable. The following example XSL style sheet, web_product_index.xsl, produces a simple HTML page by displaying content form the servable as well as the two subfragments:

```
<?xml version="1.0"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/XSL/Transform/1.0">


<xsl:template match="PRODUCTPAGE">
<HTML>
<HEAD>
<TITLE><xsl:value-of select="TITLE"/></TITLE>
<META NAME="source-xml" CONTENT="{SYSTEM/FRAGMENTID}"/>
<META NAME="source-xsl" CONTENT="web_product_index.xsl"/>
</HEAD>
<BODY>
<TABLE CELLPADDING="0" CELLSPACING="0" BORDER="0" WIDTH="451">
<TR>
<TD>
<!-- title section -->
    <FONT COLOR="#003399" SIZE="+3" FACE="Times New Roman">
    <xsl:value-of select="TITLE"/>
    </FONT><BR/><BR/>
```

```
<!-- end title section -->
<!-- short description section -->
    <B><FONT SIZE="-1" FACE="Arial">
    <xsl:value-of select="SHORTDESCRIPTION"/>
    </FONT><BR/><BR/></B>
<!-- end short description section -->
</TD>
</TR>
<TR>
<TD>
<!-- photo section -->
    <IMG  HEIGHT="110" WIDTH="140"
SRC="/multimedia{MAINPHOTO/IMAGEFRAGMENT/CONTENTDIR}{MAINPHOTO/IMAGEFRAGMEN
T/CONTENTFILENAME}" BORDER="0"
ALT="MAINPHOTO/IMAGEFRAGMENT/SHORTDESCRIPTION"></IMG>
<!-- end of photo section -->
</TD>
</TR>
</TABLE>
<!-- Highlights section -->
<TABLE BORDER="0" CELLSPACING="0" CELLPADDING="5" WIDTH="451">
<TR>
<TD COLSPAN="2" WIDTH="451">
    <B><FONT SIZE="-1" FACE="Arial">Highlights of the
    <xsl:value-of select="TITLE"/>
    </FONT></B>
</TD>
</TR>
<xsl:apply-templates select="HIGHLIGHTS/LISTFRAGMENT/LIST"/>
</TABLE>
<!-- End of Highlights section -->
</BODY>
</HTML>
</xsl:template>


<xsl:template match="/PRODUCTPAGE/HIGHLIGHTS/LISTFRAGMENT">
<xsl:for-each select="LISTITEM">
    <TR>
    <TD WIDTH="151" VALIGN="TOP">
    <FONT size="-1" face="Arial"><xsl:value-of select="TITLE"/></FONT>
    </TD>
    <TD WIDTH="300" VALIGN="TOP">
    <FONT size="-1" face="Arial"><xsl:value-of select="BODY"/></FONT>
    </TD>
    </TR>
</xsl:for-each>
</xsl:template>
</xsl:stylesheet>
```

Once all XSL style sheets for the DTDs have been defined, add them to the CHOICES list of the STYLESHEET element in the appropriate DTDs.

Then, check them in to the */xsl* directory using the Franklin Editor.

[Add here, the definition of a debug style sheet, what is should do, and where it should be saved on the server]

Step 8: Create Directory Structure

Create the directory structure that corresponds to the site map of your application. Under *publishDir* create the HTML directories, and under *multimediaDir* the multimedia directories.

If your site is published to more than one audience segment or device, define several *sibling* directory structures under *publishDir*. For example, a site published for two devices, one for browsing all content using a web browser, and another for browsing only the news section using a pda browser, could have the following directory structure:

publishDir/web/
publishDir/web/news/
publishDir/web/products/
publishDir/pda/
publishDir/pda/news/
multimediaDir/images/
multimediaDir/audio/

When an editor authors a servable, the PUBLISHDIR element of the servable will be presented in the UI with the BROWSESERVER widget. This widget allows the editor to browse the directory structure under *assetsDir* and select where to save the resulting file. Similarly, when editing a multimedia object, the widget allows the editor to browse the directory structure to select where to save the binary file.

Step 9: Configure Web Server

To browse the published sites, set up a web server for each sibling site. Configure the *Document Root* variable to point to the top of the directory hierarchy of a sibling site. The example below assumes that the *baseDir* in *franklinServletInitialization.properties* is set to "/franklin/data/":

For the example in the previous section, you would set up two web servers:

```
Web Server 1: DocumentRoot "/franklin/data/publish/web/"
Web Server 2: DocumentRoot "/franklin/data/publish/pda/"
```

Also, add the aliases below to the configuration file of Web Server 1 and 2.

```
Alias /dtd/             "/franklin/data/dtd/"
Alias /xsl/             "/franklin/data/assets/xsl/"
Alias /multimedia/      "/franklin/data/assets/multimedia/"
Alias /xml/             "/franklin/data/xml"
```

Set directory browsing "on" so that you can easily browse the DTDs and verify the uploaded XML files, XSL style sheets, and multimedia files.

Step 10: Define Roles & Users

Before running the Editor UI, you need to define the allowed roles and users of the Franklin system. A role is defined by the list of DTDs the role is allowed to edit. A user is defined by one or more roles.

To define new roles, edit the file *configDir/roles.xml* by importing the DTD *configDir/roles.dtd* to an XML editor such as Xeena from IBM alphaworks. (Future: use the Editor UI to edit the file)

The DTD *roles.dtd:*

```
<!ELEMENT ROLES             (ROLE*)>
<!ELEMENT ROLE              (ROLENAME, DTD*)>
<!ELEMENT ROLENAME          (#PCDATA)>
<!ELEMENT DTD               (#PCDATA)>
```

An example *roles.xml* defines two roles, FragmentEditor and Editor and associates the allowed DTDs to each:

```
<?xml version="1.0" encoding="UTF-8"?>
<ROLES>
    <ROLE>
          <ROLENAME>FragmentEditor</ROLENAME>
          <DTD>http://franklinserver/dtd/textfragment.dtd</DTD>
          <DTD>http://franklinserver/dtd/listfragment.dtd</DTD>
          <DTD>http://franklinserver/dtd/audiofragment.dtd</DTD>
          <DTD>http://franklinserver/dtd/videofragment.dtd</DTD>
          <DTD>http://franklinserver/dtd/imagefragment.dtd</DTD>
    </ROLE>
    <ROLE>
          <ROLENAME>Editor</ROLENAME>
          <DTD>http://franklinserver/dtd/textfragment.dtd</DTD>
          <DTD>http://franklinserver/dtd/newsarticle.dtd</DTD>
          <DTD>http://franklinserver/dtd/productpage.dtd</DTD>
    </ROLE>
</ROLES>
```

To define new users, edit the file *configDir/users.xml* by importing the DTD *configDir/users.dtd* to an XML editor.

The DTD users.dtd:

```
<!ELEMENT USERS             (USER*)>
<!ELEMENT USER              (NAME, EMAIL, PASSWORD, ROLE+)>
<!ELEMENT NAME              (#PCDATA)>
<!ELEMENT EMAIL             (#PCDATA)>
<!ELEMENT PASSWORD          (#PCDATA)>
<!ELEMENT ROLE              (#PCDATA)>
```

An example *users.xml* defines two users, each with one or more roles.

```
<?xml version="1.0" encoding="UTF-8"?>
<USERS>
```

```
    <USER>
     <NAME>Joe Moe</NAME>
     <EMAIL>joe@us.ibm.com</EMAIL>
     <PASSWORD>joe</PASSWORD>
     <ROLE>FragmentEditor</ROLE>
     <ROLE>Editor</ROLE>

    </USER>
    <USER>
      <NAME>Jane Mane</NAME>
      <EMAIL>jane@us.ibm.com</EMAIL>
      <PASSWORD>jane</PASSWORD>
      <ROLE>Editor</ROLE>
     </USER>
   <USER>
</USERS>
```

When the Franklin server is initialized, the Dispatcher module runs *Globals.loadinfo()*. This method merges *users.xml*, *roles.xml* and *dtds.xml* into one DOM in memory for fast access. The method verifies that all roles named in *users.xml* have a definition in *roles.xml*. It also verifies that all DTDs named in *roles.xml* are defined in *dtds.xml* and exist in the named directory. If any discrepancies are detected, the server prints out a warning message. (Future: the verification still needs to be implemented.)

(Future: if any of the configuration files have changed after the server was last initialized, the files will get reloaded and the DOM will get refreshed. This will not have an effect on any users currently logged on.)

Editor Interface & Dispatcher Communication

After installing the Editor User Interface application and completing the customization described in Section Install Franklin Client, launch the application from the Franklin Editor icon on the desktop.

All communications between the Editor UI and the Franklin Dispatcher follow the WebDAV protocol. [See the full specification at http://www.webdav.org/specs/]

The HTTP header of a Client request always contains the following attributes with *ACTION* replaced by PUT, GET, LOCK or UNLOCK, *franklinservername* replaced by the name of the Franklin server the client is communicating with, and *length* replaced by the length in bytes of the body section.

```
ACTION /filename HTTP/1.1
Host: franklinservername
Content-type: text/xml; charset="utf-8"
Content-Length: length
```

The body of the Client request contains the XML document to be checked into the server or a DASL search query.

The HTTP header of the Dispatcher response always contains the following attributes with *errorcode* and *message* replaced by the standard codes listed in Appendix 1.

```
HTTP/1.1 errorcode message
Content-Type: text/xml; charset="utf-8"
Content-Length: length
```

The format of the body of the Dispatcher response depends on the request and whether the request was successfully completed or not.

A special format used for the response follows the DTD below. In the further examples, the use of the elements will become obvious.

```
<!ELEMENT RESPONSE (PROCESS, STATUS, ERRORCODE, MESSAGE, SYSTEM?, LOCK?)>
<!ELEMENT PROCESS              (#PCDATA)>
<!ELEMENT STATUS               (#PCDATA)>
<!ELEMENT ERRORCODE            (#PCDATA)>
<!ELEMENT MESSAGE              (#PCDATA)>
<!ELEMENT SYSTEM           (FRAGMENTID, CREATOR, MODIFIER, CREATIONTIME,
LASTMODIFIEDTIME, PAGETYPE, CONTENTSIZE?)>
<!ELEMENT FRAGMENTIO       (#PCDATA)>
<!ELEMENT CREATOR          (#PCDATA)>
<!ELEMENT MODIFIER         (#PCDATA)>
<!ELEMENT CREATIONTIME     (#PCDATA)>
<!ELEMENT LASTMODIFIEDTIME (#PCDATA)>
<!ELEMENT PAGETYPE         (#PCDATA)>
<!ELEMENT CONTENTSIZE      (#PCDATA)>
<!ELEMENT LOCK             (LOCKEOBY, LOCKTIME, LOCKTOKEN?)>
<!ELEMENT LOCKEDBY         (#PCDATA)>
<!ELEMENT LOCKTIME         (#PCDATA)>
<!ELEMENT LOCKTOKEN        (#PCDATA)>
```

Login

The editor logs in using the user name and password defined by the Franklin administrator, as defined in Section Define Roles & Users.

Client request:

```
GET /xml/franklin_init.xml HTTP1.1
Host: franklinserver/franklinservlet
Content-Type: text/xml; charset="utf-8"

Authorization: "Basic " + encode Base64(username + ":" + password)
```

If the user name is not defined or if the password is entered incorrectly, the dispatcher responds with the appropriate error. Dispatcher response:

```
HTTP/1.1 401 SC_UNAUTHORIZED
Content-Type: text/xml; charset="utf-8"
Content-Length: length

<?xml version="1.0"?>
<RESPONSE>
  <PROCESS>login</PROCESS>
  <STATUS>401</STATUS>
```

Comment [LW4]: I don't think this is the right error code returned here. But couldn't tell cus its not working.

```
  <ERRORCODE>U101</ERRORCODE>
  <MESSAGE>User Joe Doe not defined</MESSAGE>
</RESPONSE>
```

If login succeeds, as described in Section Dispatcher: Session Management, the Dispatcher adds the user to the *currentusers* hash table and generates a unique session identifier, *sessionId*. All subsequent requests from the Editor UI must contain *sessionId* in the HTTP header.

The dispatcher responds to a successful login request by generating the *franklin_init.xml* document that corresponds to the user's roles, references the DTDs the user is allowed to edit, and specifies the attributes, operators and allowed values for the Search UI. Dispatcher response:

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset="utf-8"
Content-Length: length
Sessionid: 175a:dc8e0de306:-8000

<?xml version="1.0" encoding="UTF-8"?>
<FRANKLIN_INIT>
   <SEARCH>
     <ATTRIBUTELIST>
          <ATTRIBUTE displayname="Creation Date" name="CREATIONTIME"
class="Time"/>
          <ATTRIBUTE displayname="Last Modified Date"
name="LASTMODIFIEDTIME" class="Time"/>
          <ATTRIBUTE displayname="Creator" name="CREATOR" class="Name"/>
          <ATTRIBUTE displayname="Document Type" name="DOCTYPE"
class="Selection" options="TEXTFRAGMENT | LISTFRAGMENT | IMAGEFRAGMENT |
AUDIOFRAGMENT | VIDEOFRAGMENT | INDEXGROUP | PRODUCTPAGE | SOMESERVABLE"/>
          <ATTRIBUTE displayname="Page Type" name="PAGETYPE"
class="Selection" options="Fragment|Servable"/>
          <ATTRIBUTE displayname="Keyword" name="KEYWORD" class="Text"/>
          <ATTRIBUTE displayname="Category" name="CATEGORY"
lass="Selection" options=" Netfinity_8500R | Netfinity_7000_M10 |
Netfinity_5500_M10 | Netfinity_5600 | Netfinity_5500"/>
    </ATTRIBUTELIST>
    <CLASSLIST>
          <CLASS name="Time">
                <OPERATOR name=">="/>
                <OPERATOR name="&#60;="/>
                <OPERATOR name="="/>
                <VALUE datatype="date"/>
          </CLASS>
          <CLASS name="Integer">
                <OPERATOR name=">="/>
                <OPERATOR name="&#60;="/>
                <OPERATOR name="="/>
                <VALUE datatype="integer"/>
          </CLASS>
          <CLASS name="Name">
                <OPERATOR name="is"/>
                <OPERATOR name="isn't"/>
                <OPERATOR name="starts with"/>
                <VALUE datatype="string"/>
          </CLASS>
```

```
          <CLASS name="Text">
                <OPERATOR name="is"/>
                <OPERATOR name="starts with"/>
                <VALUE datatype="string"/>
          </CLASS>
          <CLASS name="Selection">
                <OPERATOR name="is"/>
                <OPERATOR name="isn't"/>
                <VALUE datatype="choice"/>
          </CLASS>
    </CLASSLIST>
    <RESULTS>
    <ATTRIBUTE displayname="Last Modified Date" name="LASTMODIFIEDTIME"
class="Time"/>
    <ATTRIBUTE displayname="Creator" name="CREATOR" class="Name"/>
    <ATTRIBUTE displayname="Title" name="TITLE" class="Text"/>
    <ATTRIBUTE displayname="Document Type" name="DOCTYPE"
lass="Selection"/>
    </RESULTS>
  </SEARCH>
 <ROLE name="FragmentEditor" displayname="Fragment Editor">
    <FRAGMENTS displayname="Fragment">
          <DTD displayname="Text"
href="http://franklinserver/franklin/dtd/textfragment.dtd"/>
          <DTD displayname="List"
href="http://franklinserver/franklin/dtd/listfragment.dtd"/>
          <DTD displayname="Audio"
href="http://franklinserver/franklin/dtd/audiofragment.dtd"/>
          <DTD displayname="Video"
href="http://franklinserver/franklin/dtd/videofragment.dtd"/>
          <DTD displayname="Image"
href="http://franklinserver/franklin/dtd/imagefragment.dtd"/>
    </FRAGMENTS>
    </ROLE>
  <ROLE name="Editor" displayname="Editor">
    <FRAGMENTS displayname="Fragment">
          <DTD displayname="Text"
href="http://franklinserver/franklin/dtd/textfragment.dtd"/>
    </FRAGMENTS>
    <SERVABLES displayname="Page">
          <DTD displayname="News Article"
href="http://franklinserver/franklin/dtd/newsarticle.dtd"/>
          <DTD displayname="Product Page"
href="http://franklinserver/franklin/dtd/productpage.dtd"/>
     </SERVABLES>
   </ROLE>
</FRANKLIN_INIT>
```

From this *franklin_init.xml* document, the Editor UI builds the *File* > *New Fragment* and *File* >*New Page* menus. It also maintains the mappings between display names and DTD URLs in a hash table.

The Editor UI can be set to load all DTDs at this point, if it is important to enable off-line editing. We have chosen to load a DTD from the server upon demand for a faster startup process.

At this point, the editor is able to either create new content or search for existing content in the system.

Create new content

The *File > New Fragment* menu lists all fragments and the *File > New Page* menu lists all servable pages the editor is allowed to create or edit. If the editor selects to create a new fragment, e.g. a TEXTFRAGMENT, the Editor UI gets the DTD from the appropriate URL and automatically generates a template from the DTD, as shown below:

[Include screenshot here]

In parallel, the Editor UI maintains in memory a DOM corresponding to the DTD.

Editor UI Widgets

The Editor UI uses the DATATYPE attributes in the DTD to generate the appropriate Java widget in the user interface. If an element does not contain a DATATYPE attribute no input is allowed for that element. Children elements may still contain DATATYPE attributes to specify their user interface. The mapping between Franklin UITYPEs and Java widgets are given below:

```
date          => JtextField
integer       => JtextField
string        => JtextField
shorttext     => JtextArea (scrolling)
longtext      => JtextArea (scrolling)
choice        => JComboBox(with DefaultComboBoxModel to hold the data)
browselocal   => JtextField(with JfileChooser to select local file)
browseserver  => JtextField(with custom JFrame to browse server
                    directory)
```

Each Java widget is encapsulated in a set of classes that include additional functionality. For example, if an element in the DTD is required, e.g. TITLE, the widget will be highlighted (e.g. colored brightly) to help the editor distinguish which fields must be filled in. If an element can appear more than once, e.g. KEYWORD, +/- buttons appear next to the widget that allow duplicating the widget or group of widgets.

BODY tags are handled specially within the system. The system assumes that BODY tags are composed of 1 or more PARAGRAPH tags. Typically, this is represented by a longtext widget in the user interface. Blank lines in the input are interpreted as paragraph separators. When constructing the DOM object, these paragraphs are composed within the outer BODY tag.

Check-in of New Fragment

When the editor has filled in the template in the UI, clicking on the check-in icon verifies that all required elements are filled in. If so, the DOM in memory is populated with the data in the UI widgets. New nodes are added if new instances of an element have been created using the +/- buttons. Nodes are removed from the DOM if optional fields have not been filled in. Once the DOM mirrors exactly the UI, the document is transformed to an XML string and a request is sent to the Dispatcher with the XML as the body.

Note that the only time a check-in request to the Franklin Dispatcher does not include the file name containing *fragmentid* is when uploading a new fragment. The absence of a *fragmentid* indicates to the server that a new, unnamed fragment is being checked-in. The Dispatcher assigns unique ids to all fragments.

Deleted:

Client request:

```
PUT fragmentid HTTP1.1
Host: franklinserver/franklinservlet
Content-Type: text/xml; charset="utf-8"
SessionId: 175a:dc8e0de306:-8000
Content-Length: length

<?xml version-"1.0"?>
<!DOCTYPE TEXTFRAGMENT SYSTEM
"http://adtech.ibmus2.ibm.com/franklin/dtd/textfragment.dtd" >
<TEXTFRAGMENT>
  <SYSTEM>
    <FRAGMENTID/>
    <CREATOR/>
    <MODIFIER/>
    <CREATIONTIME/>
    <LASTMODIFIEDTIME/>
   <PAGETYPE/>
    <CONTENTSIZE/>
  </SYSTEM>
  <TITLE>This is the title of this textfragment</TITLE>
  <BODY>
      <PARAGRAPH>This is the document body</PARAGRAPH>
  </BODY>
</TEXTFRAGMENT>
```

If check-in is successful the Dispatcher returns the SYSTEM data of the XML document filled in with the newly created unique *fragmentId*, and the authoring information, as described in Section XXX. The client merges the SYSTEM tag into the existing XML document. The Editor UI now has the complete XML.

Dispatcher response:

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset="utf-8"
Content-Length: length
SessionId: 175a:dc8e0de306:-8000

<?xml version="1.0"?><RESPONSE>
  <PROCESS>new Checkin</PROCESS>
  <STATUS>200</STATUS>
  <ERRORCODE>OK</ERRORCODE>
  <MESSAGE>OK</MESSAGE>
  <SYSTEM>
    <FRAGMENTID>46ba850dc8cf1f3c1007f33-Tfrg.xml</FRAGMENTID>
    <CREATOR>Joe Doe</CREATOR>
    <MODIFIER>Joe Doe</MODIFIER>
    <CREATIONTIME>2000-01-07-14.08.09.328000</CREATIONTIME>
```

```
    <LASTMODIFIEDTIME>2000-01-07-14.08.09.328000</LASTMODIFIEDTIME>
    <PAGETYPE>Fragment</PAGETYPE>
    <CONTENTSIZE>537</CONTENTSIZE>
  </SYSTEM>
</RESPONSE>
```

If the fragment about to be checked in has an accompanying content file, i.e. a multimedia asset or an XSL style sheet, the Editor UI encodes the contents using Base64encoding into the CONTENT element in the XML. On the server side, the Dispatcher un-encodes it and writes the file to the file system, as described in Section XXX. This method avoids sending multiple requests to the Dispatcher and having to maintain state between two requests.
Caveat: presently we are unclear on whether there is a size limitation to this method! And it is slow!

Check-In of Modified Fragment

If the Editor UI checks in a modified fragment or page, it will have received a LOCKTOKEN from the Dispatcher before checking it out. The check-in request in this case must include the LockTocken header field.

Deleted:

Client Request

```
PUT fragmentid HTTP1.1
Host: franklinserver/franklinservlet
Content-Type: text/xml; charset="utf-8"
SessionId: 175a:dc8e0de306:-8000
Content-Length: length
LockTocken:
```

The dispatcher verifies that the correct lock token is supplied with the PUT request before it processes the request. The dispatcher changes the MODIFIER and the LASTMODIFIEDTIME fields in the SYSTEM element.

After the check-in command, the Editor UI issues an UNLOCK command using the appropriate LOCKTOKEN.

Client request:

```
UNLOCK /12345678-tfrg.xml HTTP1.1
Host: franklinserver/franklinservlet
Sessionid  = 175a:dc8e0de306:-8000
Locktocken = 12345678
```

The Dispatcher verifies the LOCKTOKEN as described in Section XXX, and returns an OK if the token is correct, otherwise it sends one of the two Franklin lock errors:
L101 = Lock tokens do not match
L102 = Missing lock token

Check-out

To check out a fragment for editing from the Franklin Server, the Editor UI first requests a lock for the given fragment, as defined by the WebDAV protocol.

Client request:

```
LOCK /12345678-tfrg.xml HTTP1.1
Host: franklinserver/franklinservlet
Sessionid  = 175a:dc8e0de306:-8000
```

If the fragment is already locked by another user, the dispatcher returns a response with information on who has locked it when, in case the user wants to contact the person who holds the lock.

Dispatcher response:

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset="utf-8"
Content-Length: length
SessionId: 175a:dc8e0de306:-8000

<?xml version="1.0"?>
<RESPONSE>
  <PROCESS>Lock</PROCESS>
  <STATUS>200</STATUS>
  <ERRORCODE>OK</ERRORCODE>
  <MESSAGE>Locked</MESSAGE>
  <LOCK>
    <LOCKEDBY>Jane Mane</LOCKEDBY>
    <LOCKTIME>2000-01-07-13.08.10.328000</LOCKTIME>
  </LOCK>
</RESPONSE>
```

Comment [DM5]: Should not be 200 (I'll have to check)

If the fragment is not already locked and if the user with the *sessionId* is allowed to edit documents based on the DTD of the requested fragment, the dispatcher creates a unique lock on the fragment as described in section Dispatcher: Lock Management. It also sends the lock token back in the response.

Dispatcher response:

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset="utf-8"
Content-Length: length
SessionId: 175a:dc8e0de306:-8000

<?xml version="1.0"?>
<RESPONSE>
  <PROCESS>Lock</PROCESS>
  <STATUS>200</STATUS>
  <ERRORCODE>OK</ERRORCODE>
  <MESSAGE>OK</MESSAGE>
  <LOCK>
    <LOCKEDBY>Joe Moe</LOCKEDBY>
    <LOCKTIME>2000-01-07-14.08.09.328000</LOCKTIME>
    <LOCKTOKEN>12345678</LOCKTOCKEN>
```

```
  </LOCK>
</RESPONSE>
```

Now the Editor UI can request the fragment for editing using the lock received from the server.

Client request:

```
GET /12345678-tfrg.xml HTTP1.1
Host: franklinserver/franklinservlet
Content-Type: text/xml; charset="utf-8"
Locktocken = 12345678
Sessionid  = 175a:dc8e0de306:-8000
```

The Dispatcher compares the lock to the one saved in the Meta Data Store. If they match, Dispatcher responds by sending back the complete XML of the fragment.

Dispatcher response:

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset="utf-8"
Content-Length: length
Sessionid: 175a:dc8e0de306:-8000

 <?xml version="1.0"?>
<!DOCTYPE PRODUCTPAGE SYSTEM "http://franklinserver/dtd/productpage.dtd">
<PRODUCTPAGE>
   <SYSTEM>
      ...
   </SYSTEM>
   ...
</PRODUCTPAGE>
```

The Editor UI retrieves the DTD of the checked-out fragment from the server if it has not yet loaded it. Using the DTD it auto-generates the UI widgets and then fills in the existing values from the checked-out fragment, adding new instances of elements using the +/- mechanism when necessary.

The editor can modify the fields in the same way as when creating new content. Upon check-in the Dispatcher updates the LASTMODIFEDTIME and MODIFIER fields in the SYSTEM data of the checked-in XML document.

Search

Clicking on the Search icon in the Editor UI brings up the Search dialogue shown below:

[insert Search screen shot here]

When launched, the Search dialogue parses *franklin_init.xml* and stores attributes, operators and allowed values into hash tables. It dynamically generates the widgets for the query composition. When new attributes or values are added to *franklin_init.xml*, the Search code does not need to be updated.

The Search UI communicates with the Dispatcher using the Distributed Authoring Search Language (DASL) [add ref], an extension to the WebDAV protocol. Franklin Server defines the

"Franklin" name space which allows the insertion of properties that correspond to the names of the indexed elements in Franklin Meta Data Store.

An example of a DASL exchange between Search UI and Dispatcher is shown below for the example Boolean query:

```
(AND
        (PAGETYPE "is" "Fragment")
        (LASTMODIFIEDTIME "gte" "1999/10/10")
        (CREATOR "is not" "Jeff Milton"))
```

Search request:

```
SEARCH / HTTP1.1
Host: franklinserver/franklinservlet
Content-Type: text/xml; charset="utf-8"
Sessionid  = 175a:dc8e0de306:-8000

<?xml version="1.0"?>
<d:searchrequest xmlns:d="DAV:" xmlns:f="Franklin:">
  <d:basicsearch>
    <d:select>
      <d:prop>
          <f:FRAGMENTID/>
           <f:DTD/>
          <f:LASTMODIFIEDTIME/>
          <f:TITLE/>
          <f:CREATOR/>
      </d:prop>
    </d:select>
    <d:from>
      <d:scope>
        <d:href/>
        <d:depth>infinity</d:depth>
      </d:scope>
    </d:from>
    <d:where>
      <d:and>
          <d:eq>
            <d:prop> <f:PAGETYPE/> </D:prop>
             <d:literal>Fragment</D:literal>
          </d:eq>
          <d:gte>
             <d:prop> <f:LASTMODIFIEDTIME/> </D:prop>
             <d:literal>1999-10-10</D:literal>
          </d:gte>
          <d:not>
             <d:eq>
                <d:prop> <f:CREATOR/> </D:prop>
                <d:literal>Jeff Milton</D:literal>
             </d:eq>
        </d:not>
     </d:and>
    </d:where>
    <d:limit>
```

```
    <d:nresults>2</d:nresults>
     </d:limit>
   </d:basicsearch>
</d:searchrequest>
```

The Dispatcher passes the query to the Meta Data Store where the query is converted to SQL and executed against DB2. The results are converted back to the DASL format.

Currently, we are using the d:nresults tag to indicate a range of results to be returned. (e.g. <d:nresults>1-50</d:nresults>) This enables the client to request subsequent "pages" from a search with a large number of results.

Dispatcher response:

```
HTTP/1.1 207 Multi-Status
Content-Type: text/xml; charset="utf-8"
Content-Length: length

<?xml version="1.0"?>
<d:multistatus xmlns:d="DAV:" xmlns:f="Franklin:">
  <f:responsesummary>
    <f:start>1</f:start>
    <f:end>2</f:end >
    <f:total>45</f:total>
  </f:responsesummary>
  <d:response>
    <d:href>http://franklin.adtech.ibm.com/43987548-tfrg.xml</d:href>
    <d:propstat>
          <d:prop>
                <f:FRAGMENTID>43987548-tfrg.xml</f:FRAGMENTID>
                <f:DTD>Textfragment</f:DTD>
                 <f:PAGETYPE>Fragment</f:PAGETYPE>
                <f:LASTMODIFIEDTIME>2000-01-07-
14.08.09.328000</f:LASTMODIFIEDTIME>
                <f:TITLE>Lou Gerstner's bio</f:TITLE>
                <f:CREATOR/>Joe Moe</f:CREATOR>
          </d:prop>
    </d:propstat>
 </d:response>
<d:response>
    <d:href>http://franklin.adtech.ibm.com/9999999-frg.xml</d:href>
    <d:propstat>
          <d:prop>
          <f:FRAGMENTID>9999999-tfrg.xml</f:FRAGMENTID>
                <f:DTD>Listfragment</f:DTD>
                <f:PAGETYPE>Fragment</f:PAGETYPE>
                <f:LASTMODIFIEDTIME>2000-01-07-
14.08.09.328111</f:LASTMODIFIEDTIME>
                <f:TITLE>Highlights for Netfinity 8500R</f:TITLE>
                <f:CREATOR/>Jane Mane</f:CREATOR>
          </d:prop>
    </d:propstat>
 </d:response>
</d:multistatus>
```

Comment [LW6]: This example is not quite right. (ie, the numbers aren't specified this way (I don't think)

The Search UI parses the results and displays them in the results table. From the table, the editor can select items and merge them into the Active List in the Editor UI.

Future: If more than the requested number of hits exist in the database, the Search UI uses the RESPONSESUMMARY element in the result list to determine how to manage the "Next" and "Previous" buttons that allow further results or to go back to a previous set of results.

Preview

Before checking in a servable, the editor can preview the final page to be published. The preview icon in the Editor UI is active only when editing a servable. Requesting a preview sends a request to the Preview Manager servlet with the temporary contents of the servable XML. The Preview Manager expands the servable with contents of any subfragments, assembles the page with all included style sheets using LotusXSL, and returns the resulting HTML output to the client.

The Editor UI launches the web browser specified in the *franklin.properties* file and displays the temporary output. Once satisfied, the editor can check in the servable. Servables previously checked in (e.g., servables returned as search results) can also be previewed.

(Future: preview of an HTML page does not indicate to the editor which fragment produced any given area of the page. Need to devise a way to display the source element.)

Dispatcher

Session Management

When a user logs on using the Editor UI, as described in the Section Editor Interface & Dispatcher Communication: Login, the dispatcher checks for a valid user and password by consulting the DOM, generated at startup, which contains all user information. If they are valid, the dispatcher adds the user to the *currentusers* hash table and generates a unique session identifier, *sessionId*. *SessionId* is created using the java *UID()* call that guarantees to return a unique identifier for the machine the process is running on.

If the same user logs on a second time from another terminal without terminating the earlier session, the earlier *sessionId* in overwritten by a new one, and the old session becomes invalid.

At logout, the users entry is removed from the *currentusers* hash table.

Future: the class that manages the user sessions must be serializable, so that its state can be saved and reloaded if the Franklin Server servlet needs to be restarted.

System Data Creation

When a dispatcher receives the check in request from the Editor UI, it handles new and modified fragments differently.

For a new fragment, the Dispatcher fills in the following so-called SYSTEM elements in the DOM built from the incoming fragment:

```
<SYSTEM>
  <FRAGMENTID>46ba850dc8cf1f3c1007f33-tfrg.xml</FRAGMENTID>
  <CREATOR>Joe Doe</CREATOR>
  <MODIFIER>Joe Doe</MODIFIER>
  <CREATIONTIME>2000-01-07-14.08.09.328000</CREATIONTIME>
  <LASTMODIFIEDTIME>2000-01-07-14.08.09.328000</LASTMODIFIEDTIME>
  <PAGETYPE>Fragment</PAGETYPE>
  <CONTENTSIZE>537</CONTENTSIZE>
</SYSTEM>
```

FRAGMENTID is the unique identifier for the fragment. This identifier is created using the java *UID()* call which returns a guaranteed unique identifier for the machine where the process is running. To the UID, the dispatcher appends the suffix *–tfrg.xml* for simple fragments, *-bfrg.xml* for compound fragments, or *-tsrv.xml* for servables. To know which suffix to append, the dispatcher consults the *dtdToType* hash table, built at startup to cache the mapping between DTDs and their types.

CREATOR is the name of the user who originally checked in the new fragment. The dispatcher gets this name by calling the *sessionidToUser* method, which retrieves the user name from the *currentusers* hash table based on the *sessionId*.

MODIFIER is the name of the user who is currently checking in the fragment. MODIFIER and CREATOR are the same when creating the system data for a new fragment.

CREATIONTIME is the Java generated time stamp of the system data creation time at original check-in.

LASTMODIFIEDTIME is the Java generated time stamp of the system data update time at subsequent check-ins. CREATIONTIME and LASTMODIFIEDTIME are the same for a new fragment.

PAGETYPE is set to either "FRAGMENT" or "SERVABLE". The dispatcher sets PAGETYPE by consulting the *dtdToType* hash table, built at startup to cache the mapping between DTDs and their types. This field is important because processing of fragments and servables is different in the Editor UI as well as in the content store module.

CONTENTSIZE is the size in bytes of the checked-in fragment including any included binary data. The dispatcher calculates this after filling in the system data but before extracting any binary data. Thus, this is the size of the string being sent over the network between Editor UI and the dispatcher.

Name Space Management

The name space manager module of the dispatcher manages all reading and writing of files. It abstracts away the actual file system from all other modules, so that they do not have to keep track of specialized directories.

The name space manager provides the functionality to read and write into the file system DOMs, corresponding XML strings that represent fragments or servables. At dispatcher startup, the initialization file is read in, and the variables defining the directories become available to the name space manager.

When writing a compound fragment, the name space manager also extracts any encoded multimedia files and style sheets and writes them into the appropriate directories. On the other hand, when reading a compound fragment, it encodes any external files into the XML and returns the DOM to the module requesting it.

In addition to the dispatcher, the content store uses the name space manager as well. The content store uses it to read fragments from the file system and to write HTML/HDML/DHTML output files from page assembly into the file system.

The advantage of separating the name space manager from the rest of the Franklin server is to isolate the knowledge about dedicated file system directories to one module. For example, if *xmlDir* needed to be split up into a series of second-level directories to limit the size of any one directory, only the name space manager would need to know about the change. Under *xmlDir* could reside 10 child directories 0/, 1/, ...9/ and a fragment would be stored in one of them based on a load-balancing algorithm handled by the name space manager.

Coordination Between Modules at Check-in

[describe the 3 phase save to file system, meta store and tm, with the fact that tm is asynchronous. Maintenance of pending jobs table by dispatcher, and roll-back]

Lock Management

As described in the sections Editor Interface & Dispatcher Communication: Check-in and Check-out, the Editor UI and dispatcher exchange lock tokens during the LOCK and UNLOCK requests from the Editor UI.

When the dispatcher issues a lock token, it uses the Java *UID()* call to create a unique identifier. It sends the token along with the user name and the lock time to the Editor UI in the body of the response and stores another copy of this information in the meta-data store as described in the section Meta Data Store: Lock Management.

When the dispatcher receives an UNLOCK request with a lock token from the Editor UI, it needs to verify that the token matches the one stored in the meta-data store. If they match, the dispatcher issues a call to delete the lock in the meta-data store.

The dispatcher has two other ways to manage locks if problems occur. If the Editor UI requests that all locks held by the current user be released, the dispatcher issues the call *releaseLockByUser* to the meta-data store. When the dispatcher is restarted due to a system crash, all pending locks in the meta-data store can be released at startup with the call *releaseLockAll*.

Error Handling

All Franklin server side components abide to the same Franklin error handling scheme. When any of the components called by dispatcher, namely meta-data store, name space manager, user manager, or content manager, catch or throw a Java exception, they convert it to a Franklin exception and fill in all details about the context and the conditions where the error occurred.

A Franklin exception contains the following attributes:

myError - the Franklin error code
myHttpError - the HTTP error code corresponding to the Franklin error code
myMessage – explanation of error, presented to user of the Client application
myDestination – one of ERROR_USER, ERROR_LOG, ERROR_ADMIN to indicate where the error should be directed
myException – the originally caught exception, if there is one

The dispatcher module routes the exception based on the attributes. If *myDestination* is set to ERROR_USER, the dispatcher returns the exception to the Editor UI which displays the error to the user. If it is set to ERROR_LOG, the error is written to the error log file, and for ERROR_ADMIN, the process notifies the system administrator.

Meta Data Store

The meta-data store allows the indexing and searching of fragments and servables. All or a subset of the XML elements can be set to be indexed. For a large content site this allows users to quickly locate content objects of interest. The meta-data store also manages the lock information of content objects.

DB2 XML Extenders

Franklin uses XML Extenders for DB2 to index a subset of the XML elements of fragments and servables. To accomplish indexing, XML Extenders uses a Document Access Definition (DAD) that maps an XML element to a column in a DB2 table.

DB2 XML Extenders provides two different methods, namely XColumn and XCollection. We have implemented both methods in Franklin and decribe both in this document. We recommend using the XCollection as it is more flexible.

XColumn

The current XColumn implementation of XML Extenders can only map one DTD to one or more DB2 tables. In order to map all Franklin DTDs to one or more common tables, the dispatcher converts all DTDs to a so-called *universal* DTD, which contains all elements to be indexed in the set of DTDs. For this universal DTD, two DADs are created based on the XML Extenders syntax.

Two DADs are needed, because the current XColumn implementation does not support inserting elements that occur only once in the XML and those that occur more than once using a single

DAD. Thus, the examples in this section show two DADs that map values from the universal DTD.

When designing the DADs, all elements that appear only once, or single-occurrence elements, can be mapped to one table. Any elements that can appear more than once, or multi-occurrence elements, need to be mapped each into separate dedicated tables. The administrator creates a view between the single-occurrence table and the multi-occurrence tables to perform searches across all tables with one command.

A DAD specifies the following items:

```
table name = name of the DB2 table
column name = name of the column in the encapsulating table
column type = data type of the column
column path = XPath expression from the root to the element to be indexed
in the column
column multi-occurrence = flag to indicate whether the element at the path
can occur more than once
```

Example of a DAD mapping single-occurrence elements:

```
<?xml version="1.0"?>
<!DOCTYPE DAD SYSTEM "c:\dxx\franklin\dtd\universal.dtd">
<DAD>
   <dtdid>UNIVERSALDTD</dtdid>
   <validation>NO</validation>
<Xcolumn>
      <table name="META.MAIN">
      <column name="CREATOR"
        type="varchar(50)"
        path="/UNIVERSAL/SYSTEM/CREATOR"
        multi_occurrence="NO">
      </column>
      <column name="CREATIONTIME"
        type="TIMESTAMP"
        path="/UNIVERSAL/SYSTEM/CREATIONTIME"
        multi_occurrence="NO">
      </column>
      <column name="LASTMODIFIEDTIME"
        type="TIMESTAMP"
        path="/UNIVERSAL/SYSTEM/LASTMODIFIEDTIME"
        multi_occurrence="NO">
      </column>
      <column name="PAGETYPE"
        type="char(10)"
        path="/UNIVERSAL/SYSTEM/PAGETYPE"
        multi_occurrence="NO">
      </column>
      <column name="CONTENTSIZE"
        type="integer"
        path="/UNIVERSAL/SYSTEM/CONTENTSIZE"
        multi_occurrence="NO">
      </column>
      <column name="TITLE"
        type="varchar(250)"
```

```
        path="/UNIVERSAL/TITLE"
        multi_occurrence="NO">
      </column>
      <column name="DOCTYPE"
        type="varchar(32)"
        path="/UNIVERSAL/DOCTYPE"
        multi_occurrence="NO">
      </column>
      <column name="DTDURL"
        type="varchar(512)"
        path="/UNIVERSAL/DTDURL"
        multi_occurrence="NO">
      </column>
    </table>
</Xcolumn>
```

Example of a DAD mapping multi-occurrence elements:

```
<?xml version="1.0"?>
<!DOCTYPE DAD SYSTEM "c:\dxx\franklin\dtd\universal.dtd">
<DAD>
   <dtdid>UNIVERSALDTD</dtdid>
   <validation>NO</validation>
   <Xcolumn>
     <table name="META.CATEGORY">
       <column name="CATEGORY"
         type="varchar(128)"
         path="/UNIVERSAL/CATEGORY"
         multi_occurrence="YES">
       </column>
     </table>
     <table name="META.KEYWORD">
       <column name="KEYWORD"
         type="varchar(64)"
         path="/UNIVERSAL/KEYWORD"
         multi_occurrence="YES">
       </column>
     </table>
   </Xcolumn>
</DAD>
```

The tables created by these DADs are shown in the Section Table Design.

XCollection

The XCollection implementation of XML Extenders requires one DAD for each DTD to be mapped into DB2. Unlike XColumn, different DTDs can be mapped to the same DB2 tables. Thus, the XCollection implementation does not required documents to be converted to abide to one universal DTD.

[However, the current XCollection also has a few problem. We have implemented a temporary fix into the meta data store until the problems are addressed]

The DAD corresponding to textfragment.dtd is shown below:

[add DAD here]

Table Design

When a DAD is loaded into DB2, the tables and columns specified in it are automatically created. After the tables are created, the administrator needs to add the following items to the database:

- a column named ISCOMMIT in the table storing the single-occurrence elements. This column indicates if the fragment has successfully been committed to the content store and file system
- indexing on any columns that will be searched
- a view which combines data from all tables for searching

The database tables created by the previous DAD examples are shown below, along with keys, indexes, and the ISCOMMIT column created by the administrator.

```
Schema:
META:  Used for all the tables used by Franklin
INDEX: Used for all the indexes used by Franklin

Tables Generated by DAD:

META.MAIN: This table contains all the elements that occur at most once in
the input XML document
```

Column Name	Data Type	Default	Key	Index
FRAMENTID	CHAR(56)		Fk -> unifrag1	index.fragmentid
CREATOR	VARCHAR(50)			index.creator
CREATIONTIME	TIMESTAMP			index.creationtime
LASTMODIFIEDTIME	TIMESTAMP			index.lastmodifiedtime
CONTENTSIZE	INTEGER			
TITLE	VARCHAR			index.title
PAGETYPE	CHAR(10)			index.pagesize
DOCTYPE	VARCHAR(32)			index.doctype
DTDURL	VARCHAR(512)			index.dtdurl
ISCOMMIT	INTEGER	0		

```
META.KEYWORD: This table contains the KEYWORD elements associated with a
given FRAGMENTID:
```

Column Name	Data Type	Default	Key	Index
FRAMENTID	CHAR(56)		Fk -> unifrag2	index.fragmented

```
  KEYWORD              VARCHAR(64)                          index.keyword
```

```
META.CATEGORY: This table contains the CATEGORY elements associated with a
given FRAGMENTID.
```

```
Column Name          Data Type          Default       key          index

  FRAMENTID             CHAR(56)                      fk       index.fragment
                                                                 id
  CATEGORY
                     VARCHAR(128)                               index.category
```

```
UNIFRAG1: This table contains two fields,
          Fragmentid  - the fragmentid for a given XML file
          Fragmentxml - the XML file stored in the file system
The function of this table is to trigger filling META.MAIN when a record is
inserted
```

```
Column Name          Data Type          Default       key          index

  FRAMENTID             CHAR(56)                      PK
  FRAGMENTXML
                     DB2XML.XMLFILE
```

```
UNIFRAG2: This table has the same structure as UNIFRAG1. The function is to
trigger filling META.KEYWORD and META.CATEGORY when a record is inserted
```

```
Column Name           Data Type         Default       key          index

  FRAMENTID             CHAR(56)                      PK
  FRAGMENTXML
                     DB2XML.XMLFILE
```

Index

When a fragment or servable is checked in, the dispatcher converts the XML file to abide to the universal DTD for indexing in the XColumn implementation. After the conversion, it sends the universal XML to the meta-data store. In the XCollection implementation, the fragment or servable is sent as is to the meta-data store.

For XCollection, the meta-data store enters a pointer to the file into the two tables named `UNITFRAG1` and `UNIFRAG2` in the previous example. When the record is entered, a trigger copies the elements specified in the DAD to the appropriate tables. The elements are now ready for searching.

For XColumn ...

Search

When the Search UI sends a DASL search expression, described in the section Editor Interface & Dispatcher Communication: Search, to the dispatcher it passes it directly to the meta-data store. The meta-data store converts it to an SQL expression, executes the SQL query and converts the results to the DASL output format.

An example DASL query:

```
<?xml version="1.0"?>
<d:searchrequest xmlns:d="DAV:" xmlns:f="Franklin:">
  <d:basicsearch>
    <d:select>
      <f:prop>
        <f:FRAGMENTID/>
        <f:DOCTYPE/>
        <f:LASTMODIFIEDTIME/>
        <f:TITLE/>
        <f:CREATOR/>
        <f:PAGETYPE/>
      </f:prop>
    </d:select>
    <d:from>
      <d:scope>
        <d:href/>
        <d:depth>infinity</d:depth>
      </d:scope>
    </d:from>
    <d:where>
      <d:and>
        <d:like>
          <d:prop>
            <f:KEYWORD/>
          </d:prop>
          <d:literal>SERVER</d:literal>
        </d:like>
        <d:like>
          <d:prop>
            <f:PAGETYPE/>
          </d:prop>
          <d:literal>FRAGMENT</d:literal>
        </d:like>
      </d:and>
    </d:where>
    <d:limit>
      <d:nresults>1-50</d:nresults>
    </d:limit>
  </d:basicsearch>
</d:searchrequest>
```

The above DASL query converted to SQL:

```
SELECT DISTINCT fragmentID, doctype, lastModifiedTime, title, creator,
pagetype FROM meta.metaall  where  KEYWORD LIKE 'SERVER%' and PAGETYPE LIKE
'FRAGMENT' and iscommit = 1
```

An example DASL output to above query:

```
<?xml version="1.0"?>
<d:multistatus xmlns:d  ="DAV:" xmlns:f="Franklin:">
  <f:responsesummary>
     <f:start>1</f:start>
     <f:end>1</f:end>
     <f:total>1</f:total>
  </f:responsesummary>/
  <d:response>
      <d:href>http://franklinserver/46b3e60dccbcd84db007777-tfrg.xml
      </d:href>
     <d:propstat>
       <d:prop>
        <f:FRAGMENTID>46b3e60dccbcd84db007777-tfrg.xml</f:FRAGMENTID>
        <f:DOCTYPE>TEXTFRAGMENT</f:DOCTYPE>
        <f:LASTMODIFIEDTIME>2000-01-21 15:07:53.375000</f:LASTMODIFIEDTIME>
        <f:TITLE>Netfinity Highlights</f:TITLE>
        <f:CREATOR>Joe Doe</f:CREATOR>
        <f:PAGETYPE>FRAGMENT</f:PAGETYPE>
       </d:prop>
     </d:propstat>
   </d:response>
</d:multistatus>
```

If the number of results is larger than the result set requested by the Search UI, the meta-data store writes the full results into a cache file and only encodes the requested number into the DASL output. The cache file is named using an expression that encodes the query and the *sessionId* of the user. When the Search UI requests the "Next" set of results for the same query, the meta-data store does not re-execute the query. Instead it consults the cache file and extracts the appropriate next set of results. This caching scheme saves the meta-data store from executing the same query several times if the user if simply navigating within the same result set.

Note that if the contents where to change in DB2, the user does not see the updated results until he re-executes the original query without the "Next" or "Previous" flags.

Lock Management

When the dispatcher receives a `LOCK` command from the Editor UI, it creates a lock for a fragment or servable and sends the lock to the meta-data store to save in DB2. The lock information, namely `LOCKTOKEN`, `LOCKEDOWNER`, and `LOCKTIME`, is stored in the `META.LOCK` table of the following format:

Column Name	Data Type	Default	Key	Index
FRAMENTID	CHAR(56)		PK	
LOCKOWNER	VARCHAR(50)			
LOCKTIME	TIMESTAMP			
LOCKTOKEN	VARCHAR(34)			

When the dispatcher receives an UNLOCK command from the Editor UI, it issues the unlock command to the meta-data store. The meta-data store deletes the record from the META.LOCK table.

The Content Store – Daedalus (a.k.a Trigger Monitor)

This section describes how the Franklin project has extended three of the Daedalus handlers to enable the system to manage XML fragments and XSL style sheets. For the full Daedalus API documentation, read http://w3.watson.ibm.com/~challngr/papers/daedalus/index.html.

Daedalus is written in pure Java and implements *handlers* as pre-defined actions performed on the various configurable resources. Flexibility is achieved via Java's dynamic loading abilities, by more sophisticated configuration of the resources used by Daedalus, and through the use of *handler* preprocessing of input data. Most entities defined in a configuration file implement a public Java interface. Users may create their own classes to accomplish localized goals, and specify those classes in the configuration file. This permits run-time flexibility without requiring sophisticated efforts on the part of most users, since default classes are supplied to handle the most common situations.

For Franklin, we have created our own classes to implement three handlers: the Extension Parser, the Dependency Parser, and the Page Assembler.

Extension Parser

Within Franklin, Daedalus manages different types of files differently based on their extensions. Servables, simple, compound, and index fragments, style sheets and multimedia assets are all treated slightly differently in the publishing flow.

The Franklin Extension Parser takes in a name of a fragment, and returns an extension used in the Daedalus configuration files to specify actions to take during the publish process:

```
123445-trfg.xml     => tfrg  (text fragment)
123445-bfrg.xml     => bfrg  (binary wrapper fragment)
123445-ifrg.xml     => ifrg  (index fragment)
123445-tsrv.xml     => tsrv  (text servable fragment)
123445-sfrg.xml     => sfrg  (style sheet wrapper fragment)
web_index.xsl       => xsl   (style sheet)
```

The appropriate behavior for each type of fragment (e.g. source-to-sink, assemble-to-sink) is defined in the Daedalus configuration files. Generally, only servables are assembled to the sink.

Dependency Parser

The Franklin Dependency Parser reads through an XML objects that has been checked in and detects two types of dependencies:

1. Servables and fragments can include subfragments, these get stored as an edge of type "composition" in the Daedalus Object Dependency Graph (ODG).
2. Compound fragments include an associated content file, this dependency gets stored as an edge type "composition" in the ODG.
3. Servables can include style sheets, these get stored as an edge type "stylesheet" in the ODG.

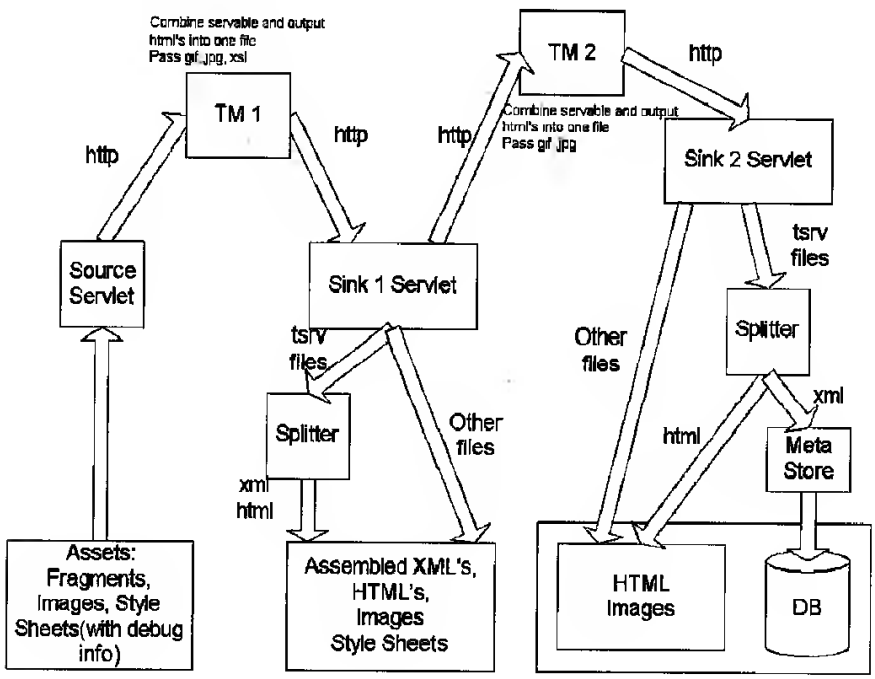
Dependencies are considered to point from the subfragments to the fragments that include them. For binary wrappers, one composition dependency points from the wrapper to the fragment that includes it, and another points from the wrapper to the binary data file that it wraps. For stylesheets, a composition dependency points from the wrapper to the stylesheet, and a stylesheet dependency points from the stylesheet to the servable that uses it.

Page Assembler

The Franklin Page Assembler expands a servable by including the contents of all included subfragments, and combines the resulting XML with the one or more style sheets using LotusXSL to produce HTML output files. The extension of each of the resulting files is determined from the stylesheet names (e.g. web_xxx_**html**.xsl). The assembled XML and all the resulting HTML files are written to one file, which is later split up in the Dispatcher, and the HTML files are written to the appropriate directories in the sink or server.

Chaining of Trigger Monitors

Currently, two Trigger Monitors are used in the publish process. They share an ODG, and the sink of the first one is the source of the second, creating a publishing chain. The following diagram shows the set-up of the Content store in its entirety:



When a fragment is checked in to the Content store, it is added to the shared ODG, and a publish command is issued to the first TM. The TM reads the fragment XML from the source servlet, uses the extension parser to find its extension, and then uses the dependency parser to find dependencies to add to the ODG. The page assembler then pulls in the contents of the fragment's subfragments, and if the fragment is a servable, combines it with its stylesheets to produce the output HTMLs. The servable XMLs, output HTMLs, binary files, and stylesheets are sent to the servlet specified as the sink of the first TM.

When a servable has been approved, a publish command on the servable fragment is issued to the second TM. It is reassembled and recombined with its XSLs, and the resulting XML and HTMLs are published to the second sink servlet. Binary files (such as images) are also published to the second sink. This is where the web server pulls the final HTML and image files from.

Example application

- managing Netfinity pages at ibm.com

Summary

Appendix 1: Error Codes

```
# Status code (101) indicating the server is switching protocols
# according to Upgrade header. (SC_SWITCHING_PROTOCOLS)

X1 = 101

# Status code (200) indicating the request succeeded normally. (SC_OK)

G200 = 200
P200 = 200
OK = 200

# Status code (201) indicating the request succeeded and created
# a new resource on the server. (SC_CREATED)

X3 = 201

# Status code (202) indicating that a request was accepted for
# processing, but was not completed. (SC_ACCEPTED)

X4 = 202

# Status code (203) indicating that the meta information presented
```

```
# by the client did not originate from the server.
(SC_NON_AUTHORITATIVE_INFORMATION)

X5 = 203

# Status code (204) indicating that the request succeeded but that
# there was no new information to return. (SC_NO_CONTENT)

X6 = 204

# Status code (205) indicating that the agent <em>SHOULD</em> reset
# the document view which caused the request to be sent. (SC_RESET_CONTENT)

X7 = 205

# Status code (206) indicating that the server has fulfilled
# the partial GET request for the resource. (SC_PARTIAL_CONTENT)

X8 = 206

# Status code (300) indicating that the requested resource
# corresponds to any one of a set of representations, each with
# its own specific location. (SC_MULTIPLE_CHOICES)

X9 = 300

# Status code (301) indicating that the resource has permanently
# moved to a new location, and that future references should use a
# new URI with their requests. (SC_MOVED_PERMANENTLY)

X10 = 301

# Status code (302) indicating that the resource has temporarily
# moved to another location, but that future references should
# still use the original URI to access the resource. (SC_MOVED_TEMPORARILY)

X11 = 302

# Status code (303) indicating that the response to the request
# can be found under a different URI. (SC_SEE_OTHER)

X12 = 303

# Status code (304) indicating that a conditional GET operation
# found that the resource was available and not modified. (SC_NOT_MODIFIED)
```

X13 = 304

Status code (305) indicating that the requested resource
MUST be accessed through the proxy given by the
<code>Location</code> field. (SC_USE_PROXY)

X14 = 305

Status code (400) indicating the request sent by the client was
syntactically incorrect. (SC_BAD_REQUEST)

THIS IS THE GENERAL (DEFAULT) ERROR RETURNED WHEN ANYTHING BREAKS

#check c102 to make sure it belongs in this area (400)
C101 = 400
C102 = 400
C103 = 400
C123 = 400
C124 = 400
D104 = 400
D110 = 400
P101 = 400
V101 = 400
F100 = 400
F101 = 400
F102 = 400
F103 = 400
F104 = 400
F105 = 400
R101 = 400
R112 = 400
R102 = 400
R103 = 400
R105 = 400
D101 = 400
D111 = 400
D145 = 400
G103 = 400

Status code (401) indicating that the request requires HTTP
authentication. (SC_UNAUTHORIZED)

G101 = 401
U101 = 401
U102 = 401
U103 = 401

L101 = 401
L102 = 401
G104 = 401
# Status code (402) reserved for future use. (SC_PAYMENT_REQUIRED)

X17 = 402

Status code (403) indicating the server understood the request
but refused to fulfill it. (SC_FORBIDDEN)

G102 = 403
D123 = 403
# Status code (404) indicating that the requested resource is not
# available. (SC_NOT_FOUND)

X19 = 404

Status code (405) indicating that the method specified in the
<code>Request-Line</code> is not allowed for the resource
identified by the <code>Request-URI</code>.
(SC_METHOD_NOT_ALLOWED)

X20 = 405

Status code (406) indicating that the resource identified by the
request is only capable of generating response entities which have
content characteristics not acceptable according to the accept
headerssent in the request. (SC_NOT_ACCEPTABLE)

F108 = 406

Status code (407) indicating that the client MUST first
authenticate itself with the proxy. (SC_PROXY_AUTHENTICATION_REQUIRED)

X22 = 407

Status code (408) indicating that the client did not produce a
requestwithin the time that the server was prepared to wait. (SC_REQUEST_TIMEOUT)

X23 = 408

Status code (409) indicating that the request could not be
completed due to a conflict with the current state of the
resource. (SC_CONFLICT)

X24 = 409

```
# Status code (410) indicating that the resource is no longer
# available at the server and no forwarding address is known.
# This condition <em>SHOULD</em> be considered permanent. (SC_GONE)

X25 = 410

# Status code (411) indicating that the request cannot be handled
# without a defined <code><em>Content-Length</em></code>. (SC_LENGTH_REQUIRED)

X26 = 411

# Status code (412) indicating that the precondition given in one
# or more of the request-header fields evaluated to false when it
# was tested on the server. (SC_PRECONDITION_FAILED)

X27 = 412

# Status code (413) indicating that the server is refusing to process
# the request because the request entity is larger than the server is
# willing or able to process. (SC_REQUEST_ENTITY_TOO_LARGE)

X28 = 413

# Status code (414) indicating that the server is refusing to service
# the request because the <code><em>Request-URI</em></code> is longer
# than the server is willing to interpret. (SC_REQUEST_URI_TOO_LONG)

X29 = 414

# Status code (415) indicating that the server is refusing to service
# the request because the entity of the request is in a format not
# supported by the requested resource for the requested method.
(SC_UNSUPPORTED_MEDIA_TYPE)

X30 = 415

# Status code (500) indicating an error inside the HTTP server
# which prevented it from fulfilling the request. (SC_INTERNAL_SERVER_ERROR)

X31 = 500

# Status code (501) indicating the HTTP server does not support
# the functionality needed to fulfill the request. (SC_NOT_IMPLEMENTED)

X32 = 501
```

```
# Status code (502) indicating that the HTTP server received an
# invalid response from a server it consulted when acting as a
# proxy or gateway. (SC_BAD_GATEWAY)

X33 = 502

# Status code (503) indicating that the HTTP server is
# temporarily overloaded, and unable to handle the request. (SC_SERVICE_UNAVAILABLE)

X34 = 503

# Status code (504) indicating that the server did not receive
# a timely response from the upstream server while acting as
# a gateway or proxy. (SC_GATEWAY_TIMEOUT)

X35 = 504

# Status code (505) indicating that the server does not support
# or refuses to support the HTTP protocol version that was used
# in the request message. (SC_HTTP_VERSION_NOT_SUPPORTED)

X36 = 505

# Error code in server.dispatcher.Dispatcher

# D104 = Error in Dispatcher.doPost()
# D110 = Fragment Type not Specified or incorrect
# P101 = Error in Dispatcher.putParseRequest()
# V101 = Error validating user

# Error codes in server.Fragment

# F100 = Error in Fragment.fragment2XML()
# F101 = Error opening Fragment.XML2fragment()
# F102 = Error parsing XML file in Fragment.XML2fragment()
# F103 = Error calling readNode("+element+")
# F104 = Error calling getElementValue
# F105 = Error calling getElementType
# F120 = Cannot close StringWriter

# Error codes in server.TextUtils

# R101 = Error in TextFile.read("+filename+")
# R112 = Error in TextFile.readTextFileWOException("+filename+")
# R102 = Error in TextUtils.createDOMfromFile("+xmlfile+")
```

```
# R103 = TextUtils.createDomFromFile SAX exception
# R105 = TextUtils.createDomFromFile IO exception
# R112 = Error in TextFile.readTextFileWOException("+filename+")


# Error codes in server.dispatcher.DomUtils

# D101 = DomUtils.documentToTuniverse TXDOM Exception
# D111 = Error in DomUtils.documentToUniversal
# D123 = Missing or invalid sessionID on checkin

# Error codes in server.dispatcher.Users

# these have destination ERROR_USER

# U101=  User + username + not defined
# U102 = Wrong password  for user + username
# U103 = User with sessionid + sessionId + not defincd

# this has destination ERROR_LOG

# U110 = Users.methodname IO exception

# Error codes in server.dispatcher.checkIn

# C103 = Error in document2String
# C102 = Checkin Error
# C101 = Users.checkProvledge error

# DE111 = Delete Error

# G200 = successful get
# P200 = successful put

# Locking errors
# L101 = Lock tokens do not match
# L102 = missing lock token


#MISC
# F108 = invalid FragmentID
# C123 = error in fragment2XML
# C124 = Failed saving content to metadata store

# G104 = Authorization String Empty
# D145 = error parsing input stream
```

Evaluation of Franklin and Kittyhawk Integration Concept

Please use the scenarios to assess the value of an integrated version of Kittyhawk and Franklin Editor. After you complete each scenario, please answer the corresponding questions.

In your assessment of the value of the Franklin concept, please remember there may be UI problems and bugs in this pilot implementation. Try to assess the value of the concept rather than the value of this current implementation.

Use the role Editor in FranklinRole field in your user profile. To create projects and assign tasks, you need to be assigned the Kittyhawk role Administrator.

Note that for Scenarios 1-3 you are playing the role of a Regular User. For Scenarios 4-5, you are playing the role of a Superuser. Before beginning tasks 1-3, make sure your Franklin role is KittyHawk User profile is set to "Editor" and Superuser to "No". Change to the Superuser to "Yes" for tasks 4-5.

I called these A, B, C etc. Just to distinguish from existing ones. These will be renumbered 1,2,3 after we decide which ones to

Scenario A: Create ar ***gments***

Set Superuser to "Yes" a

Assume that: You are the *› create and publish images and thumbnails for the Ti* *›motions, news and product pages. Other editors will l* *:ate at least one of each.*

See the definitions of IMA
http://franklin.adtech.inte

B

[Here we should give the *:e and publish a fragment.*
Or should we make this o *ruser yet????*
I wanted this to be differe *that Scenarios build on each other, does that mak*

Scenario B: Assign and Complete a "Create and Publish" Task for a Product Spec fragment

Set Superuser to "No" and Franklin Role to "Editor" in KH

Assume that: Enterprise Sites need to host Product information on the latest model in the ThinkPad line. This work needs to be delegated to an editor responsible for creating product specs.

See the definition of PRODUCT SPEC at
http://franklin.adtech.internet.ibm.com/franklin/downloads//ESiteDTDs.html

[This would be essentially Scenario 1 tailored to a Product Spec]

Scenario C: Assign and Complete a "Create and Publish" Task for a Product Page

Set Superuser to "No" and Franklin Role to "Editor" in KH

Assume that: Enterprise Sites needs to host a series of pages on the latest model in the ThinkPad line. The product spec, images and thumbnails have been uploaded earlier, now the tasks of creating and publishing a Product Page for the new ThinkPad need to be delegated to an Esite editor. This page is to be published for the web only.
See the definition of PRODUCT PAGE and PRODUCT COMPARISON at
http://franklin.adtech.internet.ibm.com/franklin/downloads//ESiteDTDs.html

[this would be scenario 2 done once for PRODUCT PAGE]
[questions afterwards would emphasize that product spec, image and thumbnail in first two scenarios were reused and did not have to be recreated]

Scenario D: Assign and Complete a "Creote" task ond a "Publish" Task for a Product Comporison
Set Superuser to "No" and Franklin Role to "Editor" in KH
Assume that: Enterprise Sites needs to host a page compariong the latest model in the ThinkPad line with older models to highlight it's advantages. The product spec, images and thumbnails have been uploaded earlier, now the task of creating a Product Comparison needs to be delegated to an Esite editor and the task of reviewing the work and publishing it to a QA person. This page should be published for the web only.
See the definition of PRODUCT COMPARISON at
http://franklin.adtech.internet.ibm.com/franklin/downloads//ESiteDTDs.html

[this would be scenario 2 done once for PRODUCT COMPARISON but with separate tasks for Create and Publish]
[questions afterwards would emphasize that product spec, image and thumbnail in first two scenarios were reused and did not have to be recreated]

Scenorio E: Assign and Complete a "Edit and Publish" Task for a Product Page
Set Superuser to "No" and Franklin Role to "Editor" in KH
Assume that: The Product Page on the latest ThinkPad created in Scenario C needs to be published for customers with a PDA and a notification sent to customers with Smart Phones. This task needs to be delegated to an Esite editor.
See the definition of PRODUCT PAGE and PUBLISHINFO at
http://franklin.adtech.internet.ibm.com/franklin/downloads//ESiteDTDs.html

[this would be scenario 2 done once for PRODUCT PAGE this time with Edit+Publish task]
[questions afterwards would asl about the overhead of publishing for PDA and phone.

Scenorio F: Identify a problem on the Esites web sites ond request that it be fixed
Set Superuser to "Yes" and Franklin Role to "Editor" in KH
Assume that: While browsing the Esite, you see a bad typo on the Product Page created in Scenario C. As a Superuser, you bring the page into Franklin, fix the typo, republish, and verify that all affected pages were republished.

See the definition of PRODUCTSPEC at
http://franklin.adtech.internet.ibm.com/franklin/downloads//ESiteDTDs.html

[this would be a brand new scenario where user starts by browsing Esites, sees a problem, copies and pastes the URL into Franklin File-> Retreive by Publish URL, checks out the page, checks out the imbedded PRODUCT SPEC subfragment, fixes the typo, checks in, reviews all affected pages, republishes the PRODUCT SPEC, and then goes back to Esites to see that the page has changed. I can write the step for this one because it includes some new funcationality that you may not have tried yet]

Scenario 1: Assign and Complete a "Create and Publish" Task for a Fragment
[this would be incorporated into A and B]

#	Steps	Expected Results
1	*KittyHawk Steps: Use Administrator Role* *1. Create a Request* *2. Create a Project and associate the Request to it* *3. Create a CREATE AND PUBLISH task* *4. ASSIGN the task to a Franklin Editor with Regular User role (Note: Assign the task to yourself so you can complete it using Franklin Editor)* *5. Provide a DESCRIPTION of the document to be created* *6. Designate a DTD name (e.g. Thumbnail, Form, Product Spec, or Form)* *7. SEND the task* *8. Save and close the task, and project*	*1. A new task with Status "Sent" should be in the task section on the project.* *2. Check that the task is in the KittyHawk editor queue.*
2	*Franklin steps: Use regular Franklin User (Editor) role* *1. Launch the Franklin UI and Login* *2. Get tasks assigned to you* *3. Start the task to create the appropriate fragment, fill it in , and check it in* *4. Approve the final pages (and note that there are no pages to approve because no servable page includes the fragment you just created)* *5. Publish the fragment*	*1. In Franklin, after publish, fragment should disappear from the Active List.*
3	*KittyHawk Steps: Use Administrator Role*	*1. Task should have Status "Comp".* *2. Fragment ID should be filled in for the task.*

	1. Open the project with previously assigned task 2. Click Refresh button above Task Section	3. 'URLs of work' field should be filled on the project farm. (If fragment is not used in any servable pages yet, the message should say "No URLs to view. This fragment is not used in any final page") 4. Check that the task is no longer in the KittyHawk editor queue.
4	View task in Project farm. Since all tasks on the project are complete, you can now click the Begin Final Approval process button.	Requesters will be notified that the work has been complete.

Scenario 1 Questions:

How satisfied are you with the Kittyhawk/Franklin process for completing this scenario?

Very Dissatisfied	Dissatisfied	Neutral	Satisfied	Very Satisfied
1	2	3	4	5

Please explain.

If this scenario includes tasks you perform, how does Kittyhawk/Franklin compare to the current method/tool you use? Kittyhawk/Franklin is. . .

Much Worse	Worse	About the Same	Better	Much Better	N/A
1	2	3	4	5	

Please explain.

Scenario 2: Assign and Complete a "Create and Publish" Task for a Servable Page, use Save as Draft in Franklin

[this would be replaced and incorporated into C]

Repeat the same steps used in Scenario 1, only this time create a servable page by selecting one of the following: promotion, productpage, productcomparison or link.

All other steps are the same as in Scenario 1, except:

#	Steps	Expected Results
2	*Franklin steps: Use regular Franklin User (Editor) role* *1. Launch the Franklin UI and Login* *2. Get tasks assigned to you* *3. Verify that the "Create+Publish" task has no FRAGMENTID associated with it.* *4. Start the task to create the appropriate servable document* *5. Search for subfragments to include in the document* *6. Cut and paste the desired fragments into the document* *7. Select style sheets for the web, the pda, and the Slingshot index page.* *8. Preview your work in between edits* *9. Complete filling in the document* *10. Check in the document as a DRAFT* *11. Remove it from the Active List* *12. Refresh tasks assigned to you* *13. Verify that the "Create+Publish" task now has a FRAGMENTID associated with it* *14. Start the task again* *15. Make further edits* *16. Check document in* *17. Verify that document appears highlighted in Active List* *18. Approve final pages* *19. Publish document*	*1. Saving as Draft should update the "Create+Publish" Task with the fragmentID*

Scenario 2 Questions:

How satisfied are you with the Kittyhawk/Franklin process for completing this scenario?

Very Dissatisfied	*Dissatisfied*	*Neutral*	*Satisfied*	*Very Satisfied*
1	2	3	4	5

Please explain.

If this scenario includes tasks you perform, how does Kittyhawk/Franklin compare to the current method/tool you use? Kittyhawk/Franklin is. . .

Much Worse	*Worse*	*About the Same*	*Better*	*Much Better*	N/A
1	2	3	4	5	

Please explain.

Scenario 3: Copy and Paste Fragments from Franklin into a Kittyhawk Project

[this would stay as is]

#	Steps	Expected Results
1	*Log into Franklin.*	
2	*Click on the search button in left panel. Give query parameters, and search for a list of Fragments.*	
3	*Select several fragments. Click copy button.*	*Copies fragment information to clipboard.*
4	*Open KittyHawk. Open an existing project, or create a new project.*	
5	*Click PASTE FROM FRANKLIN button.*	*Creates a task for each fragment.*

Scenario 3 Questions:

How satisfied are you with the Kittyhawk/Franklin process for completing this scenario?

Very Dissatisfied	Dissatisfied	Neutral	Satisfied	Very Satisfied
1	2	3	4	5

Please explain.

If this scenario includes tasks you perform, how does Kittyhawk/Franklin compare to the current method/tool you use? Kittyhawk/Franklin is. . .

Much Worse	Worse	About the Same	Better	Much Better	N/A
1	2	3	4	5	

Please explain.

Scenario 4: Search and try to check out documents in Franklin
[this would stay as is]

#	Steps	Expected Results
1	Log into Franklin.	
2	Click on the search button in left ponel. Try out different combinations of seorch ottributes, operators and values. Verify results.	
3	Select a document in the Search results and try to check it out.	Check out icon should be disabled.
4	Select a document in the Search results and check out for view only.	Document should appeor in Right ponel in Read only mode.
5.	Try to check in document displays in right hand panel	Check in icon should be disobled.
6	Try to check out document displayed in right hand panel.	You should not be able to check it out, as it has not been assigned in a task to you..
7.	Remove document displayed in the right hand panel from the Franklin Editor	

Scenorio 4 Questions:

How satisfied ore you with the Kittyhawk/Fronklin process for completing this scenario?

Very Dissotisfied	Dissatisfied	Neutral	Satisfied	Very Satisfied
1	2	3	4	5

Please explain.

If this scenario includes tasks you perform, how does Kittyhawk/Franklin compare to the current method/tool you use? Kittyhawk/Franklin is. . .

Much Worse	Worse	About the Same	Better	Much Better	N/A
1	2	3	4	5	

Please explain.

Scenario 5: Assign and Complete an "Edit" task and a "Publish" task for a Promotion

[this would be replaced and incroporated into Scenario D because it's two separate tasks, and Scenario E because it's an Edit, not a Create]

#	Steps	Expected Results
1	KittyHawk Steps: Use Administrator Role 1. Create a Request 2. Create a Project and associate the request with it 3. Initiate a task to EDIT a Promotion servable 4. Click on SEARCH FRANKLIN button to get a fragment ID from Franklin 5. ASSIGN the task to a Franklin Editor with Regular User role (yourself in this case) 6. Send the task 7. Save and close task 8. Initiate a task to PUBLISH the same Promotion servable 9. ASSIGN the task to a Franklin Editor (yourself so that you can complete the task - in reality you would assign it to a different person than the previous EDIT task) 10. Close task without sending. 11. Close the project	1. A new EDIT task with Status "Sent" should be in the task section 2. A new PUBLISH task with Status "New" should be in the task section 2. Check that the EDIT task is in the KittyHawk editor queue
2	Franklin steps: Use regular Franklin User (Editor) role 1. Launch the Franklin UI and Login 2. Select Tasks -> SHOW TASK INTERFACE, and view tasks assigned to you 3. Start the task to check-out the Promotion. 3. Edit the Promotion and check it in 4. Select SHOW TASK INTERFACE to verify that task has disappeared.	1. In Fronklin, the Promotion appeors highlighted in Active List. 2. The Check-out, Approve and Publish icons should be disabled for the Promotion, as you have not been assigned the PUBLISH task at this point. 3. Task should not appear in Task Interface anymore.
3	KittyHawk Steps: Use Administrator Role 1. Open the project with previously assigned tosk. 2. Click Refresh button above Task Section. 3. Edit the PUBLISH task, send it, and close task.	1. EDIT tosk should have Status "Comp". 2. 'URL of work' field should NOT be filled for the EDIT task on the project form. 3. Check that the EDIT task is no longer in the KITTYHAWK editor queue. 4. PUBLISH task should have Status "Sent"
4.	Franklin steps: Use regular Franklin User (Editor) role 1. Select Tasks -> SHOW TASK INTERFACE, and view tasks assigned to you	1. Starting the PUBLISH task should launch the browser with links to all final pages to preview.

	2. Start PUBLISH task to begin approval of final pages. 3. Review all pages in the browser 4. Publish the Promotion 5. Exit Franklin Editor	
5.	KittyHawk Steps: Use Administrator Role 1. Open the project with previously assigned task 2. Verify that all tasks are complete 3. Click on "Begin Final Approval" 4. Copy and paste appropriate URLs of work to Request document.	1. PUBLISH task should have Status "Comp". 2. 'URL of work' field should be filled in for the PUBLISH task on the project form. 3. Check that PUBLISH task is no longer in the KITTYHAWK editor queue.

Scenario 5 Questions:

How satisfied are you with the Kittyhawk/Franklin process for completing this scenario?

Very Dissatisfied	Dissatisfied	Neutral	Satisfied	Very Sotisfied
1	2	3	4	5

Please exploin.

If this scenorio includes tasks you perform, how does Kittyhowk/Franklin compare to the current method/tool you use? Kittyhawk/Franklin is. . .

Much Worse	Worse	About the Same	Better	Much Better	N/A
1	2	3	4	5	

Please exploin.

Scenario 6: Assign a "Publish" Task, Create and Address Problem Report
[we could include the creation of a problem report in the "publish" part for Scenario D or should we leave it as a separate scenario as is so that the scnarios are shorter and address fewer functionalities? What do you think?]

Instead af creating an "Create" task far this ane, assume that a praduct page has already been created.

#	Steps	Expected Results
1	*KittyHawk Steps: Use Administrator Role* *1. Open an existing project* *3. Initiate a task to PUBLISH a Product Page servable* *4. Click on SEARCH FRANKLIN button to get a fragment ID from Franklin* *4. ASSIGN the task to a Franklin Editar with Regular User role (yaurself in this case)* *5. Send the task* *6. Save and clase task, and project*	*1. A new PUBLISH task with Status "Sent" should be in the task section* *2. Check that the task is in the KittyHawk editor queue*
2	*Franklin steps: Use regular Franklin User (Editor) role* *1. Launch the Franklin UI and Lagin* *2. Select Tasks -> SHOW TASK INTERFACE, and view tasks assigned to yau* *3. Start PUBLISH task to begin approval of final pages.* *4. Review pages in the browser* *5. Create a problem report for the web document* *6. Double click on one of the praduct data elements in the table.* *7. Fill in the PROBLEM REPORT and send.* *8. Clase the brawser* *9. Remove Product Page with problem from your Active List* *10. Verify that the PUBLISH task still appears in our Task interface.*	*1. Starting the PUBLISH task should launch the browser with links to all final pages ta preview.* *2. After clicking on "Create Problem Repart", you shauld see the fragmentid and element name change in the browser status bar as you mouse over the different areas of the page.* *3. Double-clicking on an area should launch a Problem Report form to fill in.* *4. In KittyHawk, task should still appear as "Sent" as it was never campleted due ta problem.*
3	*KittyHawk Steps: Use Administrator Role* *1. Open the project with previously assigned task.* *2. View the Problem Report* *3. Assign a new "Edit and Publish" task with fragmentid from Problem Report (assign it to yourself) and the problem stated*	*1. Problem Report shauld appear in the Project.*
4.	*Franklin Steps: Use regular Franklin User (Editor) role to complete the "Edit and Publish" task as usual*	

5. *KittyHowk Steps: Use Administrator Role*
   1. Refresh tosk list
   2. Note that "Edit ond Publish" task is completed
   3. Edit the "Publish" task still uncompleted due to Problem Report
   4. Reclick on SEND to resend the tosk to the same editor (to notify him that problem has been corrected)
6. *Franklin Steps: Use regular Franklin User (Editor) role to complete the "Publish" tosk as usuol.*

1. When approving final poges, the cause of the Problem Report should be fixed.

Scenario 6 Questions:

How satisfied are you with the Kittyhawk/Franklin process for completing this scenario?

Very Dissatisfied	*Dissatisfied*	*Neutral*	*Sotisfied*	*Very Satisfied*
1	2	3	4	5

Please explain.

If this scenario includes tasks you perform, how does Kittyhowk/Franklin compare to the current method/tool you use? Kittyhowk/Fronklin is. . .

Much Worse	*Worse*	*About the Same*	*Better*	*Much Better*	N/A
1	2	3	4	5	

Pleose explain.

Scenario 7: Create a Superuser Activity Log and View Conflict Reports as Superuser

[this would stay as is also]

Change your Superuser flag to "Yes" in your KittyHawk User Profile

#	Steps	Expected Results
1	*Open KittyHawk. Create a project.*	
2	*Assign several EDIT tasks associated with one Fragment ID.*	
3	*Assign all to regular Franklin role editors.*	
4	*Send Tasks, Save Project, and close KittyHawk.*	
5	*Franklin Steps as a Superuser:* *1. Launch Franklin Editor* *2. Select "File->Check out by Fragmentid" to check out the FragmentID that you assigned to other editors in Step 2* *3. View tasks assigned to other editors for the document in the Conflict Report.* *4. Click on "OK" to check out anyway* *5. Edit the document* *6. Check it in and view the Conflict Report again.* *7. Click on "OK" to check in anyway.*	*1. Conflict report should warn user of other active tasks on the same fragment.*
6.	*KittyHawk Steps: as Project Administrator* *1. Open the project navigator.* *2. Open the project.* *3. Click on the doclink.*	*1. There is a lightning bolt icon in the view to indicate that there was 'superuser activity' on one of the tasks related to the project.* *2. There is a doclink in the Superuser Activity field. The history on the project also indicates that there was superuser activity on the project.* *3. The superuser activity log is opened.*
7	*As a regular editor, check out a fragment that is in a task not assigned to you.*	*Not allowed because the user cannot see them.*
8	*As a Superuser, check out a fragment that is not assigned in the other tasks.*	*The fragment is checked out without any warnings.* *The Conflict Report will not appear at check out or check in of this fragment until it is assigned to a task.*
9	*Edit fragment and check it in.*	*A superuser activity log is created in Kittyhawk.*
10	*Repeat test by assigning tasks to other Franklin roles in KittyHawk: Image Editor, Fragment Editor.*	

Scenario 7 Questions:

How satisfied are you with the Kittyhawk/Franklin process for completing this scenario?

Very Dissatisfied	*Dissatisfied*	*Neutral*	*Satisfied*	*Very Satisfied*
1	2	3	4	5

Please explain.

If this scenario includes tasks you perform, how does Kittyhawk/Franklin compare to the current method/tool you use? Kittyhawk/Franklin is. . .

Much Worse	*Worse*	*About the Same*	*Better*	*Much Better*	N/A
1	2	3	4	5	

Please explain.

Note that for Scenarios 8-9 you are playing the role of a Superuser. Before beginning, make sure your Franklin role is KittyHawk User profile is set to "Editor" and Superuser to "Yes.

For the pilot we are using 3 Enterprise Sites: Cargill, North Carolina, and Bayer. You must register to each one separately, with a different user name, because the registrations cannot be shared between different E-Sites. (If you choose to register only at one, make sure you publish content to that E-Site in the Scenarios below)

Scenario 8: Register with the Enterprise Sites, publish ond view o Product COmporison at an E-Site.
[this scenario should come eorlier, so that they register to Esites, moybe should be after Scenarios A ond B, the first time they hove to actuolly publish a servoble poge, not just o fragment. I think this should be changed ta be A PROMOTION instead af Product Comporisan because we already have o Praduct COmparisan in Scenaria D...
]

#	*Steps*	*Expected Results*
1	*E-Site registration steps:* *1. To register, go to* http://amadeus.sby.ibm.com/servlet/gold/NorthCarolina/Welcome *2. Click on "Register Now".*	*After logging on to one E-Site, your browser is set with a cookie for that particular E-Site. The cookie lasts for the current browser session. To register or logon to a different E-Site, you need to launch a new browser and go to the URL of that site. The 3 E-Site URLs are:*

3. Follow instructions to register until you are brought back to the Login screen of the Enterprise Site you started from.
4. Enter the IBM ID and password you just created.
5. When prompted far the IBM authorization code, enter "nc100" (You will receive this code in an e-mail ot some point, this gets you started faster)
6. On the home page of the E-Site, click on "Edit Personalization"
7. Select o Job Type and Check ALL available options (to ensure that regardless of interest area classificatian of a document, it will appear far you)
8. Click on "Submit", then click on "Return to your IBM home"
9. Note the different sections of the site: Home, Product, News, Solutions, etc.
10. Close the browser.

11. Launch a new browser and repeat steps 1-10 for the other 2 Enterprise Sites:
http://amadeus.sby.ibm.com/servlet/gold/Cargill/Welcome
http://amadeus.sby.ibm com/servlet/gold/Bayer/Welcome

2 *Franklin steps:*
1. Launch the Franklin UI and Login.
2. Create a Product Comparison. To ensure that you will see the document on the E-Site under your profile:
- *enter START_DATE as today or earlier to make the article appear immediately*
- *select one or more ENTERPISE Sites you want it to appear under*
- *select the LOCATION, or the section of the site you want it to appear under*
- *select a few INTEREST_AREAS*
- *do not select HOME_FEATURE (it is not working for now)*
- *select the JOB_TYPE you set in your E-Site profile or "All"*

3. Search far Product Specs and Thumbnails and include one of each in the Product Comparison using Copy and Paste.

http://amadeus.sby.ibm.com/servlet/gold/NorthCarolina/Welcome
http://amadeus.sby.ibm.com/servlet/gold/Cargill/Welcome
http://amadeus.sby.ibm.com/servlet/gold/Bayer/Welcome

4. Create a newThinkPad Product Spec (to include as the second product in the Product Comparison) and check it in.
5. Copy and Paste the new Praduct Spec fram the Active List ta the Product Camparisan.
6. Preview the harizontal and vertical style sheets and select the one you prefer.
7. Check in the Product Comparison.

When trying to check in the Product Comparison, a dialogue should alert you that a subfragment has not been published.

8. Check out the unpublished subfragment using the small check out icon ta the right of it in the Product Comparison.
9. Check the fragment back in, then publish it from the Active List.
10. Try to check in the Product Comparison again.

When trying to check in the Product Comparison, no dialogue should pop up this time.

11. Approve the final pages for the Product Comparison.
12. Publish the Product Comparisan.

3. *Enterprise Site steps:*
   1. Logan to one of the E-sites where you published the Product Comparison.
   2. Go to the section you selected under Step 2 for LOCATION.
   3. Locate the Product Comparison and view the document.
4. *Franklin steps:*
   1. Search and check out the Praduct Comparison
   2. Change the LOCATION tag.
   3. Check in the Product Comparison.
   4. Publish the Product Comparison.
5. *Enterperise Site steps:*
   1. Go to the section that correspands to the new LOCATION yau specified in Step 4.
   2. Refresh the page in the browser.
   3. Locate the Product Comparison.

Product Comparison should not appear under old LOCATION, and shauld naw appear under new LOCATION. There may be a slight delay, so try a few times if you do not see the change immediately.

6. *Franklin steps:*
   1. Check out the Product Spec you created in Step 2.4.
   2. Edit the PRICE_DOLLARS field.
   3. Check it in, and Publish.
7. *Enterprise Site steps:*
   1. Locate the Product Comparison.

The republish af the Product Spec should have triggered the republish of the Product

2. Refresh the page. 3. Verify that it reflects the updated price.	Comporison. There may be o slight delay, so try a few times if you do not see the change immediately.

Scenario 8 Questions:

How satisfied are you with the Kittyhawk/Fronklin process for completing this scenario?

Very Dissatisfied	Dissatisfied	Neutral	Satisfied	Very Satisfied
1	2	3	4	5

Please explain.

If this scenario includes tasks you perform, how does Kittyhowk/Franklin compare to the current method/tool you use? Kittyhawk/Franklin is. . .

Much Worse	Worse	About the Same	Better	Much Better	N/A
1	2	3	4	5	

Please explain.

Scenario 9: Verify the locking of documents in Franklin

[this should stay as is]

Complete this scenario with a collaegue, both of you as Superuser.

#	Steps	Expected Results
1	Franklin steps: User 1: 1. Lounch Franklin Editor. 2. Search for a document of your choice. 3. Check it out.	Document appears in right hand panel.
2	User 2: 1. Launch Franklin Editor. 2. Search for the same document as User 1 in Step 1. 3. Check it out.	You should get a dialogue stoting that the document is locked by User 1. Choose to check it out for "Read only".
3	User 1: 1. Remove the document from the right hand panel.	Document is highlighted in Active List, but no longer checked out.

4	*User 2:* *1. Check out the document in the right hand panel.*	*You should not get the lock message, and document should check out.*
5	*User 1 & 2:* *Exit Franklin Editor.*	

Scenario 9 Questions:

How satisfied ore you with the Kittyhawk/Franklin process for completing this scenario?

Very Dissatisfied	*Dissatisfied*	*Neutral*	*Satisfied*	*Very Satisfied*
1	2	3	4	5

Please explain.

If this scenorio includes tasks you perform, how does Kittyhawk/Franklin compare to the current method/tool you use? Kittyhawk/Franklin is. . .

Much Worse	*Worse*	*About the Same*	*Better*	*Much Better*	N/A
1	2	3	4	5	

Please exploin.

Current State			Action			Return	Next State		
TaskID	IsCommit	IsDraft	Method	TaskID	AsDraft	value	TaskID	IsCommit	IsDraft
Create a new draft									
Null			Create	Xxx	1	True	Xxx	1	1
Create a new fragment by checkin									
null			Create	Xxx	0	True	Xxx	0	0
Update a draft with a draft									
Xxx	1	1	Update	Yyy	1	True	Yyy	1	1
Update a draft with a checkin									
Xxx	1	1	Update	Yyy	0	True	Xxx	9	1
							Yyy	0	1
Update a checked-in fragment with a dr									
Xxx	1	0				ue	null	1	1
Update a checked-in fragment with anot									
Xxx	1	0				ıe	Xxx	9	0
							Yyy	0	0
Commit a newly checked-in fragment									
Xxx	0	0	C				null	1	0
Rollback a newly checked-in fragment									
Xxx	0	0	R				null		
Commit an updated fragment									
Xxx	9	0	Co				null	1	0
Yyy	0	0							
Rollback an updated fragment									
Xxx	9	0	Rollback			Yyy	null	1	0
Yyy	0	0							
Commit a checked-in never committed fragment									
Xxx	9	1	Commit			Yyy	null	1	0
Yyy	0	1							

Rollback a checked-in never committed fragment									
Xxx	9	1	Rollback			Yyy	null	1	1
Yyy	0	1							
Query whether a checked-in yet uncommitted fragment that initially was a draft can be published									
Xxx	9	1	CanPublish	Yyy		M_NRDY	Xxx	9	1
Yyy	0	1					Yyy	0	1
Query whether a checked-in yet uncommitted fragment can be published									
Xxx	9	0	CanPublish	Yyy		M_NRDY	Xxx	9	1
Yyy	0	0					Yyy	0	1
Query whether a checked-in committed fragment can be published									
Xxx	1	0	CanPublish	Yyy		M_OK	Yyy	1	0
Query whether a draft can be published									
Xxx	1	1	CanPublish	Yyy		M_DRFT	Xxx	1	1

How to Install Franklin Editor

1. Create a **franklin** directory in a location of your choice on your hard drive. These instruction assume you create **C:\franklin**

2. Download the self-extracting FranklinEditor from http://monolith.adtech.internet.ibm.com/franklin/downloads/FranklinEditor.exe. Save the file to a temporary directory on you hard drive.

3. Double-click on **FranklinEditor.exe** in the temp directory on you hard drive. This will start the WinZip Self-Extractor. Under "Unzip to folder" enter **C:\franklin** or the path to the franklin directory you created in Step 1. Click on "Unzip".

4. Go to **C:\franklin\FranklinEditor** on your hard drive. Open **franklin.properties** file in NotePad. If the location of your browser is different from the one listed, remove the **#** in front of **browserPath** and change the value to the appropriate path on your machine. Save and close the file.

5. Double-click on **C:\franklin\FranklinEditor\Fran[illegible]or.bat** file. The login screen will popup and prompt for your username and[illegible] team has provided to you.

How to Dele

1. Simply delete the drive.

How to Get Started with Franklin Editor

After installing the Editor by following the How To Install instructions and logging in, you can take the actions listed below. When Franklin runs integrated with the KittyHawk workflow engine, a user can work in one of two modes: **superuser** or **regular user**.

Note:
All icons in the Editor UI display a tooltip if you mouse over the icon.
Useful error messages appear in the status bar at the bottom of the Editor UI.

ACTIONS FOR SUPERUSER

Search

Click on the "Search" icon above the Active List. This brings up the Search UI. Select attributes and values from the drop down menus. You can add more search conditions using the +/- widget. Click on "Submit" to launch the search.

Hint: A search that will always bring back results is "Page Type is Fragment"

Once search results are displayed in the table, you can select one or more of them and merge them into the Active List in the Editor UI by clicking on the "Merge with Active List" icon

Check Out for Edit

Select an item in the Active List and click on the "Checkout selected document" icon. If the document is not locked by another user, it will appear in the right-hand pane in editable widgets. You can modify any fields and resubmit into Franklin server. See "Check in"

Check Out for View

Select an item in the Active List and click on the "View selected document in read only mode" icon. It will be displayed in the right-hand page in editable widgets. However, you will not be able to check-it in with any changes. You can click on the "Check out" icon above the right-hand pane to check it out for edit.

Create New Fragment

Click on the "Create new document" icon. This brings up the list of fragments and pages you are allowed to create. Select a fragment, click on "Create". The Editor UI retrieves the correspondig DTD from the server and auto-generates the right-hand pane with widgets. The required fields are highlighted in yellow, and must be filled in before you are allowed to check in the document.

Create a New Page

Create a page the same way you create a fragment. However, note that a page has additional fields that

enable it to be turned into a final HTML page and previewed:

A page includes one or more subfragments. To include a subfragment into a page under construction, do the following:

- Search for all subfragments of the appropriate document type (see Search), and merge them into your Active List
- Select the subfragment you wish to include from the Active List
- Click on the "Copy" icon above the Active List 
- Click on the subfragment field in the page under construction
- Click on the "Paste" icon above the right hand pane.  This will write the fragment ID of the pasted fragment into the field.

A page requires "PublishInfo" to be filled in. You must select a publish directory on the server. This is where the final page will be saved. You must also enter the final HTML file name for the published page. You must also select a style sheet to render the page in HTML.

See "Preview" to view the final HTML page.

 Check in

Once you have created a new document or modified an existing one, click on the "Check in document" above the right-hand pane. The document will be validated against the DTD and sent to the Franklin server. You can now search for it to check it out again for modifications.

 Preview a Page

Before checking in a page, you can preview it by clicking on the "Preview page" icon above the right-hand pane. It will launch the browser you specified in the *franklin.properties* file during installation. The browser will display the output HTML generated using the style sheet listed in the *first* PUBLISHINFO of the page. To see all output pages, you need to check-in the page and then click on the "Approve" icon.

Note that you can make further changes to the page and preview it again before checking it into the server. You can also preview any page (but not a fragment) by selecting it from the Active List and clicking on the "Preview page" button above the Active List.

 Approve document

To approve the publishing of a fragment or a servable, select it in the Active List and click on the "Approve final pages for selected document" icon. This will launch a browser and display a list of all resulting HTML pages. For a fragment, the list consists of all final HTML pages of all the servables that include the fragment as a subfragment. For a servable, the list consists of all final HTML pages of that servable.

 Create problem report

If you find a problem with a final page, create the appropriate problem report by clicking on the "Create

problem report" icon.

 Publish document

If you find no problems with any of the final pages you are approving, click on the "Publish document" icon above the Active List. The selected document will be published to the server.

 Remove current document

While editing a document in the right-hand pane, you can click on the "Remove current document" icon. This will unlock the document on the server and discard the document being edited from the Editor UI.

ACTIONS FOR REGULAR USER

Get Task List

Select "Tasks -> Show Task Interface" from the menu bar to retrieve current tasks assigned to you in the workflow engine.

 Update Task List

To refresh the entries in the Task Dialogue, click on the "Update task list" icon. Note that after you first launch the Task Dialogue the tasks do not get automatically updated. You have to explicitly ask for the list to be updated.

 Initiate Task

To begin working on an assigned task, select the task and click on the "Initiate selected task" icon. A Create task will open a new document template to fill in, an Edit task will check out an existing document, and a Publish task will launch a browser to approve pages to be published. Once you check-in or publish the document initiated by a task, the task will disappear from the task dialogue.

 View task info

To view task associated with a document in the Active List or in the right hand panel, click on the "View task associated with selected document" icon. It will bring up the Task Dialogue with the appropriate task selected. Note that this icon is only enabled for documents that are associated with a task.

The other actions you will be able to take in the Franklin Editor UI as a regular editor are described above in the superuser section. Allowed actions will be identified by the icons being highlighted.

Franklin User Acceptance Testing

Participant Feedback Analyzed for the Franklin Team by
Roger Tilson
IBM Ease of Use Architecture and Design
July 17, 2000

E

Executive Summary

Participants generally liked the functionality that Franklin provided, and they were generally satisfied with the tool overall. Participants liked that Franklin

- *Publishes data in as many different formats as desired*
- *Solves the problem of data maintenance on the Web*
- *Stores product data in XML*
- *Provides the obility to publish content without help from developers*
- *Provides the obility to change content once and hove the changes oppeor in multiple ploces*
- *Provides the ability to convert product doto to non-Web platforms*
- *Provides the ability to preview*
- *Allows sharing of fragments*
- *Provides better organization of content/data via standardization*
- *Allows the user to click around the site and easily change the page*
- *Allows the user to retrieve documents based on URL*

In addition to these generolly positive comments, porticipants noted oreos for improvement. In porticulor, participonts expressed dislike for the current UI, or what they called "getting around" in it. They recommended, either explicitly or implicitly, several minor changes, such as right-click options, double clicking to open/checkout documents, keyboard shortcuts for copy and paste, a more conspicuous icon for the search interface, and a way to sort lists of items in the search interface by clicking on headers. They also recommended or implied that some major changes would be valuable. Specifically, participants suggested enabling users to browse the Web and identify/select servables and fragments for editing, ond creoting a browsoble librory (distinct from the seorch interface) of frogments and servobles that also enobles previewing.

The Franklin team needs to implement the minor changes participants recommended, and also consider some of the major changes. The magnitude of the UI design changes the team undertakes will likely depend upon the goals/requirements for Franklin. If the goal is for users to be as efficient as they can be using Franklin, and to learn it as quickly as possible, then the Franklin team needs to gather more user input to determine the optimal UI design for interacting with servables and fragments. If the goal is only for users to be more efficient thon they ore currently, then severol minor chonges to the UI will likely suffice. The usobility goals for Fronklin will dictate whether more user input and mojor design changes are necessary.

Recommendations

The most importont recommendation involves completing a design walkthrough. The specific recommendations for improving the UI appear in two categories, one for major design changes, and one for minor changes.

Complete a design walkthrough or head-to-head comparison

If the goal is for users of Franklin to be as efficient as they can be, then the Franklin team needs to complete a design walkthrough showing users different possible designs for finding and working with servables and fragments. Among these different possible designs would be those of the competition. The main goal of the walk through is to determine which design(s) works best. Other goals are to determine if there is a reason or odvontage in continuing to develop o new product, whot those

advantages are, and whether the new product being considered needs changes to the conceptual design. The different designs used can be paper sketches, screen mockups, or a fully functional tool like Franklin is currently. Whichever they are, participants "walk through" accomplishing particular tasks. See the UCD site for more information on design walkthroughs (w3.ibm.com/ucd).

Consider the following major UI changes:

- *Additional views (e.g. tree diagrams, or other mechanisms determined by user input) for finding, checking out, and previewing fragments and servables*
- *A feature that allows users to browse the site and find the page/fragment they want, and then select and open it from the browser to edit it*
- *A mechanism allowing users to organize servables and fragments according to their needs*
- *A preview function in the search interface so users can determine if a fragment or servable is the one they want (perhaps previewing the selected fragment or servable in a right pane while the left shows the list of fragments or servables)*

Make as many of the following UI changes as time and resources permit:

- *Provide a short tutorial explaining how to get started using Franklin*
- *Give default focus to the user name field of the Franklin logon interface (also enable keyboard use to logon)*
- *Provide easy-to-understand labels for fields in templates*
- *When possible, change to standard Lotus or Microsoft icons, or icons that users are more familiar with*
- *Use text labels with all icons or those icons that may be unfamiliar*
- *Make it more obvious that the search button is active when Franklin first launches*
- *Enable use of keyboard for all functionality, especially copying and pasting fragments*
- *When users access the directory, open to the location users were last viewing*
- *When the Franklin window is resized, adjust the size of option and entry fields so that the entire UI fits into the window; establish a minimum size for entry fields, at which point horizontal scrolling is required*
- *Do not close the draft when users click save as draft (not sure what all users expect save as draft to do, but one person recommended this, use extra discretion here)*
- *Provide messages that not only tell users there is a problem, but tell users how to solve the problem*
- *Provide messages to indicate why preview will not work in some situations*
- *Provide localized help to explain the function of specific fields in templates, or a prominent link to a page showing examples of how the data is used*
- *Indicate for all fields what information is and/or is not necessary (e.g. whether adding a $ sign is necessary in price fields, and whether adding the abbreviation MB is necessary for memory fields)*
- *Indicate beside the name fields that users need to add file extensions such as .jpeg or .gif*
- *Add right-click functionality, such as for copying and pasting fragments, or checking out fragments and servables*
- *Enable users to sort search results by creator, dates, etc., by clicking on the metadata headings*
- *Ensure user ids and logins aren't case sensitive, and that users get the same search results when they type roger tilson or Roger Tilson as the creator*
- *To check out fragments, allow users to type names in addition to copying and pasting them*

- *Facilitate double-clicking to open/checkout documents and create new ones*
- *Provide more cues in the search interface as to the status of use of fragments and servables: are they currently checked out, and are they currently published on the site*
- *Provide a way to publish to multiple servers*

Caveats

This user input will be most valuable as a means to improve the UI and functionality rather than as an assessment of the value of Franklin. It will not be very useful as an assessment of the value of Franklin for the following reasons:

- *At least two participants thought Franklin was for product data only, which caused them to rate Franklin lower on key scales*
- *One participant did not realize that Franklin was intended to be used as part of a workflow process, and that they did not use this aspect of the product*
- *The esites meta data was complex and foreign to this different group of users, which made the tasks difficult to complete*
- *Participants reported that sometimes the instructions were not clear or contained irrelevant information, and most could not complete task 8 because the document was not checked out as was intended*

If the Franklin team still wants a proof-of-concept user evaluation, then Franklin will need to be customized to meet the specific needs of the user group that performs the evaluation. The scripts will also need to be pilot tested, since some of the instructions were inaccurate.

Participants

Four out of six of the participants currently create or maintain content for Web sites. The other two participants are involved in determining which tool(s) the TG group will use to create and manage Web content.

Findings

Participants liked the functionality that Franklin provided. Specifically, they liked that it:

- *Publishes data in as many different formats as desired*
- *Solves the problem of dato maintenance on the Web*
- *Stores product data in XML*
- *Provides the ability to publish content without help from developers*
- *Provides the ability to change content once and have the changes appear in multiple places*
- *Provides the ability to convert product data to non-Web platforms*
- *Provides the ability to preview*
- *Allows sharing of fragments (with other content providers? Or documents? Or both?)*
- *Provides better organization content/data via standardization*
- *Allows the user to click around the site and easily change the page*
- *Allows the user to retrieve documents based on URL*

Below are the ratings for three of the post-test questions, and the comments of the participants:

Overall satisfaction

How satisfied are you <u>overall</u> with the Franklin process for completing these scenarios?

Very Dissatisfied	*Dissatisfied*	*Neutral*	*Satisfied*	*Very Satisfied*
		1	*4*	

Mean Avg: 3.8 (Between Neutral and Satisfied)

Lynn (Neutral): I liked the way it captured the data and how that data could be used anywhere in as many different formats as someone wanted. Getting around the tool was difficult.

Michelle (Satisfied): I believe that Franklin is a good tool to complete a lot of the content management tasks. However, the main comment I have is that the instructions for the test are not clear, making it difficult for me to evaluate Franklin. The 30min orientation (given the down times) gives only a cursory view of the tool. I would be able to give better feedback if I understand it more.

Dave (Satisfied): Although not impressed in comparison to our other tools, Franklin is a good product. It gets the job done, is fairly easy to use after being trained and learning how the UI works, and solves a legitimate problem with data maintenance on the Web.

Getting started

How easy or difficult was it to get started using Franklin?

Very Difficult	*Difficult*	*Neither Difficult Nor Easy*	*Easy*	*Very Easy*
	1	*1*	*3*	

Mean Avg: 3.4 (Between Neither/Nor and Easy)

Lynn (Neither/Nor): When the login screen appears, the top text box should have a set focus on it.

Patty (Easy): The hardest part was understanding the meta data for the e-sites.

Phyllis (Difficult): I did not know about the icons to the right of the screen (i.e. check out subfragment or add fields).
I did not know that fragments or subfragments had to be 'merged' onto the active list in order to use them.

Dave (Eosy): It is not a difficult tool. There were frustrations at the beginning, however. For example, wanting to close the current document, and not finding where the close button was. (Closing the entire application instead.) Also, Copy and Paste did not seem to work, and other functions that are normally taken for granted in any production application.

Do you think Franklin needs to be easier to get started using?

Lynn: Yes. There could be a tutorial provided.
Patty: No
Phyllis: Yes
Michelle: Yes, An average user may not be well versed in "common" software navigation. Need more comprehensive training before use and continued support during use. Would be useful to have a help manual (local and Web).

Dave: Yes. Things as simple as using standard icons for close, copy, paste, etc. would be a great help. Although the tool is not hard to learn, the questions are screaming in my mind of why the programmers made up their own icons for copy/paste, among other things?

Ease of using once learned

How easy or difficult was using Franklin once you had learned how it worked?

Very Difficult	Difficult	Neither Difficult Nor Easy	Easy	Very Easy
			2	2

Mean Avg: 4.5 (Between Easy and Very Easy)

Phyllis (Very Easy): The tool was easy to use after I had received help from Dikran. In the future, the eMeeting should be allotted more time to ensure that the introduction may be completed.

Michelle (Easy): Once you understand how it works, it's easy though there are little quirks here and there.

Dave (Easy): As mentioned, the initial learning curve is quick, then the tool is easy to work with. The exception to this is the product page form, which is way too complex for the average user.

Comparison with other tools

How does Franklin compare to the current method/tool you use to manage the content of Web sites? Franklin is. . .

Much Worse	Worse	About the Same	Better	Much Better	N/A
	2				3

Mean Avg: 2 (Worse)

Lynn (Worse): Worse than our new tool. We would have to set up extensive training on the Franklin tool and then dedicate resources to be a pseudo help desk.

Dave (Worse): Although Franklin has some added features, it is missing many more. It seems to be more a tool to just manage the XML for product data than for true content management. Also, the UI features need quite a bit of work to make the tool workable for most users. If Franklin could be integrated as a part of a complete content management system, it would add a good deal of value.

Task efficiency

Tasks users can complete more efficiently using Franklin:
Lynn: Viewing data under different environments, Storage of XML data

Phyllis: The ability to publish content without help from developers
The ability to change content once even though it is located in multiple places

Dave: Product Spec sheets, conversion of product data to non-Web platforms

Tasks users can complete more efficiently using other tools:
Lynn: Our tasks are easier to perform. Our new tool will have the ability to modify non-product data.

Phyllis: None

In general, would Franklin allow you to complete your tasks more efficiently :
Lynn: Franklin would allow our team to perform a fraction of our tasks more efficiently. We still have the overview page, news, support, press releases and a few other templates.

Phyllis: Yes. It will reduce the need for help from developers. Content will, consequently, be updated or modified more frequently.

Dave: No. Most of IBM does not have a robust content management tool, and would get great value from Franklin. However, PSD has a tool already, and is developing the next generation of that tool. The ideal solution would be to integrate the strengths of Franklin with the rest of their tool.

Franklin advantages

Lynn: Good way to store XML data.

Patty: The ability to edit Fragments and their meta data. Also, the ability to preview.

Phyllis: The ease of publishing content.

Michelle: Self-service tool for content providers.
Allows sharing of fragments.
Better organization of pages/content via standardization.

Dave: Storage of Product data in XML, available to both Web and non-Web platforms from the same data source.

Franklin disadvantages

Lynn: It seems to be limited in entering data only for products. How does an administrator create new style sheets or adjust current ones?
The UI needs some work, but we were told not to take that into consideration (a little hard since we are trying to use the tool).

Patty: The user interface needs some improvement, i.e. Colors, help features...Would also like to be able to copy an existing fragment/servable and customize to new content.

Phyllis: Franklin does not have workflow capabilities.

Michelle: The UI is not very friendly. Need to provide extensive training to users.
Conversely, the "better organization" of content also means that there is lesser flexibility.

Dave: Lack of UI. It is not intuitive to use, and therefore requires support and customization for any group that wants to use it. The fact that Franklin is already being used, yet we are going through this exercise to evaluate it for TG, is a perfect example of its need to be improved in UI and flexibility.

Would you want to use Franklin or similar process/tool?

Lynn: Yes
Patty: Yes. However, I am concerned about our content providers. They currently use a home grown interface that does not require them to fill in meta-data. Perhaps (just brainstorming) we'd need a layer on top of this for those who want to provide content but just use the existing meta data values and therefore don't even show them.
Phyllis: Yes. At this point there is a lengthy turnaround time for content changes since developers are given the task of loading content as opposed to content managers/authors.
Michelle: Yes
Dave: Yes. Franklin is definitely on the right track. And mentioned previously, if integrated with a more complete content solution, it would be a valuable tool.

What would you most like to change?

Lynn: The user interface needs to be a little bit more helpful. I would also like to have seen the administrator's pt of view. How does a team create new style sheets?

Patty: I know we didn't use the workflow part, but I'd like to be able to have a baseline for content, so when updates are made the reviewer can see what exactly has changed without reading the entire piece of content.

Michelle: Friendlier UI. Better navigation.

Dave: User Interface

Scenario 2

Very Dissatisfied	Dissatisfied	Neutral	Satisfied	Very Satisfied
		1	5	

Mean Avg: 3.83

Comments:
Carl (Neutral): With or without a content management tool, what we need is a well-ordered, well-maintained, user-friendly image library, and when new images are created we need content owners to put them into the library. Adopting Franklin (or any other tool) will not automatically cause this to happen.

Lynn: (Satisfied): It allowed me to complete the task. "Content" field should be above "Content FileName". When a user browses for an image, they can find it in the local directory then the file name just appears in the "Content" field. When browsing for the image, if the user selects one, and it is wrong, they have to browse for it again. When they hit that button to browse, the directory is not where the user last looked, it is in the Franklin Tool directory. I think it should remember where the user looked last.

Patty (Satisfied): Don't think that I, the user, should have to check for duplicate name before creating the image. Also, please provide some filename help...i.e. Naming guidelines.

Phyllis (Satisfied): I was pleased with the tool AFTER I had help from Dikran. For example, in the CONTENTFILENAME field I did not know to add a file extension (i.e. jpg) from the error message 'could not map filename null'. A suggestion for tool improvement is to allow double-clicking on fields. For example, upon creating a fragment or page, I was hoping to double-click on the fragment type to create it. Instead I am forced to click on the CREATE button on the bottom.

Michelle (Satisfied): Easy to use. Would be great if Franklin can "remember" where I last pulled my files from. "Content File Name" - why can't this be pulled from the file name of the gif/jpg automatically?

Dave (Satisfied): The form for submitting images is fairly simple and straightforward. It is easy to work with, and being able to view the directory structure on the server is a nice touch. However, the UI is not intuitive, and needs work.

Scenario 3

Very Dissatisfied	Dissatisfied	Neutral	Satisfied	Very Satisfied
	2	3	1	

Mean Avg: 2.83 (Between Dissatisfied and Neutral)

Comments:

Carl (Dissatisfied): First: I couldn't figure out how to get out of "Thumbnail" mode, so I had to shut down Franklin and restart it.
Second: I had to re-edit franklin.properties before it would preview the file. The requirement to configure franklin.properties will be a huge barrier to most of our prospective content "owners" because by and large they are not technical people. With content coming from many people in many locations, I see this as a significant barrier to its successful widespread adoption. Rather than bottom-line content "owners" uploading content, we will likely end up with a few people on the web team doing it.
Third: The Scenario instructions ask you to "refresh tasks assigned to you". I never did figure out how to find out what tasks were "assigned to me". I finally ignored this instruction.
Fourth: When I edited the product page and saved it as a draft, my edits disappeared from the fields on the right side of Franklin and the downlevel version reappeared. This is very counter-intuitive. Keeping the downlevel version is great, but when you save your edits you should continue to see the new version. Each time I checked the document in, my edits would disappear and the downlevel version would reappear.
Fifth: When I checked out the product page, and then did a search, the version I checked out did not appear in the search results. It should appear, with a notation that it has been checked out.

Lynn (Neutral): I'm assuming it is a product specialist involved with the creation.

Patty (Neutral): Insufficient help with error messages.

Michelle (Neutral): Navigation is difficult especially between search window and main window. "Right click" functions would help.
The instructions are not clear. Some steps don't seem relevant.
Would be very helpful if each meta data has a detail/brief description. Maybe this is not so bad for someone who knows the product well. I am not well versed, so I have difficulty.
Got an error message - "Automation server cannot create object." after Step #18. Dikran said it's a security problem.

Dave (Dissatisfied): The form is too complex for end users. The people who write the content for product pages are not technical. Based on our experiences with the first generation of our content management tool, if the forms are too complex, even with training, the system just won't get used. The process of filling out the form and submitting is fine, but the form needs to be simplified, and the terminology on each field has to be written in English, not the field names that make sense to the system programmers.

Scenario 4

Very Dissatisfied	*Dissatisfied*	*Neutral*	*Satisfied*	*Very Satisfied*
		3	*2*	

Mean Avg: 3.4 (Between Neutral and Satisfied)

Comments:

Lynn (Neutral): The tool wasn't checking in at all. I tried to modify all the fields and they didn't work until I cut out the registered symbol in the summary. Then it checked-in fine. The exact error was, " an error parsing input stream".

Phyllis (Neutral): After I had checked the page in I had gatten the message 'can nat preview this page'. Does this have something to do with the style sheet for PDA?

Michelle (Neutral): Instructions not clear. Followed instructions but the PDA link did not show up at the "approval pages" stage. Dikran tried it and the PDA link showed up on his pc.

Dave (Satisfied): The process works, and if the page is already created, the form does not seem as daunting as when creating a page from scratch. However, had I not already been shawn how to add the PDA style sheets, I would have had trauble figuring it out on my own, and the style sheets/layaut are not visually separated from the rest of the fields on the form.

Scenario 5

Very Dissatisfied	*Dissatisfied*	*Neutral*	*Satisfied*	*Very Satisfied*
		2	2	1

Mean Avg: 3.8 (Between Neutral and Satisfied)

Comments:
Lynn (Satisfied): The general thought of capturing data like this is great. The UI is a problem. Assuming the person entering in the info is a product specialist, it still doesn't specify whether MB should be entered for memory or just a number.

Phyllis: (Neutral) I had tried to preview my fragment but every time I had clicked on the preview icon nothing happened.

Michelle (Neutral): No explanation of each meta data. A product specialist might know but I am not well versed. Had some difficulties understand the fields required.

Dave (Satisfied): Again, aside from UI complaints, the tool does the job intended.

Scenario 6

Very Dissatisfied	*Dissatisfied*	*Neutral*	*Satisfied*	*Very Satisfied*
		1	4	

Mean Avg: 3.8 (Between Neutral and Satisfied)

Comments:
Lynn (Satisfied): I think the purpose behind it is great. The product is good at enabling the user to complete this scenario. But w/o directions, I would have been lost. I used product specs created by other people that were already published. I didn't understand why they needed to be checked-in/published again for my task.

Phyllis (Satisfied): The tool is excellent. I just had difficulty adding a product spec since my screen did nat display the '+' sign to the right of the field. Withaut asking Dikran, I wauld nat have known to scrall to the right of the screen to click on the '+' sign.

Michelle (Neutral): Suggest thot the "price" meta dota field indicate that the "$" is default. If nat, end up with values like "$$3500".

My thumbnail did not shaw up. Dikran explained that it is in the index page and not product comparisan page.

My Praduct Comparisan page did not shaw up. The page olso did not show up in the seorch function. Recreated the pages twice using different nomes (reploced "&" with "_" becouse of XML) . Still did nat show up. "The requested URL/web/ProdCompA&T.html was not found on this server."

"Caps" or "na caps" far user name/creator field. I logged on Friday under "Michelle Lim". I lagged on Monday under "michelle lim". When I do searches by creator, I get different results when I use "Michelle Lim" and "michelle lim". It's canfusing.

Scenario 7

Very Dissatisfied	*Dissatisfied*	*Neutrol*	*Sotisfied*	*Very Satisfied*
			3	*2*

Mean Avg: 4.4 (Between Satisfied and Very Satisfied)

Comments:
Lynn (Satisfied): Allaws the user to click around the site and easily change the page.

Patty (Very Satisfied): I'm extremely impressed with the functionolity to retrieve based on URL.

Phyllis (Sotisfied): This functionolity is excellent. The only difficulty I had encountered was checking out the subfragment. I did not see the icon to the right of the praduct spec field. I kept copying the fragment id af the subfragment and going to 'FILE CHECK OUT WITH FRAGMENT ID', which did not change the right-side of the screen.

Michelle (Very Satisfied): This part is easy. =)

Scenario 8

Very	*Dissatisfied*	*Neutral*	*Satisfied*	*Very*

Dissatisfied *Satisfied*

Comments:
Lynn (Dissatisfied): I couldn't just type the id in the field. I had no conflict report or it wasn't apparent as to where it was.

Phyllis: I could not type the fragmentid into the appropriate field. I did not receive an error message stating that another was using the field either. Hence, I could not complete this task.

Michelle (Neutral): The check out by fragment ID window.... only accepts a cut and paste of the ID. Does not allow direct entry into the box.
Did not get a message that the fragment is locked but a conflict report did come up. However, was not given the option to click "OK" and check out anyway (Step #4). Could not evaluate.

Dave: N/A – There were no tasks in the system, so this scenario did not function as the test described

Scenario 9

Very Dissatisfied *Dissatisfied* *Neutrol* *Satisfied* *Very Satisfied*

Comments:
Lynn: I had no one to work with on this.

Phyllis: There were no colleagues to test this with. Hence, I did not complete this scenario.

Michelle: Could not evaluate with another user.

Dave: N/A – I was working alone, so could not test with another user.

Discussion

The Franklin team may want to establish short-term and long-term goals for the Franklin content management system. In the short-term, the team could provide many of the fixes that participants in this evaluation recommended. In the long-term, the team could examine different possible views and mechanisms for interacting with servables and fragments, and provide the optimal solution. Providing the smaller fixes will increase user sotisfoction, and the ease of completing tasks. The long-term work will address the underlying or deeper causes for the participants disliking the current UI, and will likely make a bigger impact on increasing ease of use and user satisfaction.

Short-term fixes can help users learn the interface, and minimize problems if users' conceptual model differs from the model upon which Franklin operates. For instance, the search button is the only button operable in the initial view of Franklin. This button, however, is gray and appears too similar in state to the other buttons that are inoperable at this point. Providing a better visual cue that users can begin work by clicking the search button, and implementing other improvements that users in this evaluation recommended, can go a long way toward improving the usability of Franklin.

That all of the participants listed the UI as the primary disadvantage of the product suggests that these short-term fixes will not arrive ot the optimol solution for interacting with servables and fragments, however. More work needs to be done to arrive at a solution that matches the user's conceptual model.

A few of the bigger issues that need to be addressed are:

- *What is the base view for interacting with servables and fragments?*
- *What is the start view for interacting with servables and fragments, and is it different from the base view?*
- *Do users want/need additional views or mechanisms for interacting with servables and fragments?*
- *How do users conceptualize organizing fragments and servables?*
- *How do users conceptualize accessing servables and fragments?*
- *How do users conceptualize moving from a view of documents in a library to a work view?*

Franklin currently opens into the work-plane view in which no documents are visible to beginning users. This view could also be considered the base view. One reason users may find Franklin initially difficult to use is because much of the functianality is initially hidden. There are not many cues for how to begin. An alternative design solution would be to open into a library or browsable view of documents provides a search interface for finding and checking out fragments and servables. It may be that users would prefer the library of documents, and the ability to preview documents, to be the base view.

Currently, users enter parameters and search for documents they want to work with, or they can retrieve documents based on the URL. The search functionality and retrieve based on a URL are both very useful tools, but users might like additional functionality and additional views of the content. Currently, users cannot see from the search interface how documents are organized. Providing a browsable library could be one means of providing an overview of the page types, or servables, and the fragments that constitute the pages. The library, in the form of a simple tree structure for example, could facilitate accessing fragments by the servables that contain them, which in turn could give users additional cues as to which fragments are used to create specific servables.

codeReview

- positioning for deeply indented dtds (ie not in iv_mainPane)

- I noticed one oddity when using the +/-: I added 2 list items to listfragment and i only filled in linktitle and description. i checked it in, then checked it back out. now when i add more list items using +/- , it only adds linktitle and description, not the 4 fields that should be there. do you know what might be going on? i can show you on Monday if this description is cryptic...

- CheckoutForView (like checkout without lock)
  implement checkoutForView which is a GET without lock token
  flag on fragment, iv_readOnly = false;
  from either active list or search can do a getForView
  - put into edit window (right side) without being able to modify
    disable the checkin button for read only fragments
  - title bar include READ only "-- Read only --"
  [weird cases:
   [1 viewing fragment, now want to check it out, can't easily do it]
   [2 if viewing fragment, then search and do a checkout, it will just go to existing fragment, which is still in read mode]
   [3 test... with new icons]
  ]

- Logout... pr ents or keep for editing
  choose which
  offline edit file, locktoken
  on login fil ragments
  1. unlock unv
  2. save dtd &

- offline editi local dtds.

- stale session ORIZED

- show media fi owseLocal entry

- publish info l operly indented

- copy and creat

- search nresult

- captitalize fo... (and smaller) for table headings...

- (tag1 | tag2)* method hasQualifier doesnt find model with () groupings

- chuck button for audio fragment, checkOutForView

- change login init file name ie "franklin_init.xml" -> "/login"
  synch with jeff

Scott Smiley

From: Jon Gibbons
Sent: Monday, April 04, 2005 4:43 PM
To: Scott Smiley
Subject: FW: Franklin Editor UI

From: Dikran S Meliksetian [mailto:Dikran_Meliksetian@us.ibm.com]
Sent: Thursday, March 24, 2005 2:30 PM
To: Jon Gibbons
Cc: Louis Weitzman; Sara Elo-Dean
Subject: Fw: Franklin Editor UI

Jon,

Here is a note dated ...m to download/install and use the franklin client.
We are still looking at ...elieve should be in November/December 1999 timeframe. The a... robust working code and manuals taht could be delivered to pc...

Dikran

---------- Forwarded by **Sara Elo/Armonk/IBM**

02/02/2000 03:49 PM

...ountain View/IBM@IBMUS, John ...omers/IBM@IBMUS, Stephen Kennedy/Somers/IBM@IBMUS, Patrick Rooney/Boulder/IBM@IBMUS

cc Franklin, Maria Hernandez/Somers/IBM@IBMUS

Subject Franklin Editor UI

Hi all,

Install:
Please go to http://monolith.adtech.intemet.ibm.com/franklin/downloads/index.html for
"How to download Franklin Editor UI" and
"How to get started with Franklin UI"

Once you have the client installed and running you might want to try the following things:

Try out:
1) Search for "Document Type = SoftwareSalesManual", Check-Out from the server the fragment entitled "Net.Commerce". This is based on the DTD Patrick provided, filled in with Net.Commerce data. Feel free to create a new SWSalesManual fragment to see how the authoring works.

2) Search for "Document Type = SWSALESMANUALPAGE" and select the "Net.Commerce" servable you get back. This is a page that imports the fragment in 1) If you merge this document into your Active List, you can preview it to see how a page gets rendered into HTML using LotusXSL. You can also check it out to see its contents.

3) Search for "Document Type = PRODUCTPAGE". Check-Out from the server the "Netfinity" servable and Preview it. This is another example of a Page made up of 4 subfragments rendered in HTML.

4) Go wild, create any fragments and pages to get a feel for the editor. The current server is a play space

Caveats:
- Please ignore anything entitled "Test" in the search results. Those are our tests...
- In the SWSalesManual, the fields containing <P>, <LI> etc... are not yet rendered correctly in the final HTML. We are working on integrating the mechanism, small matter of programming...

User Names:
Below are your usernames and passwords for the Franklin Editor:

```
<USER>
    <NAME>Ron Lautmann</NAME>
    <EMAIL>lautmann@us.ibm.com</EMAIL>
    <PASSWORD>ron</PASSWORD>
    <ROLE>Editor</ROLE>
  </USER>
<USER>
    <NAME>John Dorval</NAME>
    <EMAIL>dorval@us.ibm.com</EMAIL>
    <PASSWORD>john</PASSWORD>
    <ROLE>Editor</ROLE>
  </USER>
<USER>
    <NAME>Stephen Kennedy</NAME>
    <EMAIL>stephen@us.ibm.com</EMAIL>
    <PASSWORD>stephen</PASSWORD>
    <ROLE>Editor</ROLE>
  </USER>
<USER>
    <NAME>Patrick Rooney</NAME>
    <EMAIL>rooney@us.ibm.com</EMAIL>
    <PASSWORD>patrick</PASSWORD>
    <ROLE>Editor</ROLE>
  </USER>
</USERS>
```

Don't forget that tooltips exist for all icons....
Comments, bug reports welcome, please send to me and i'll pass on...

Regards,
Sara

.....
Advanced Internet Technology

http://w3.webahead.ibm.com

Scott Smiley

From: Jon Gibbons
Sent: Monday, April 04, 2005 4:43 PM
To: Scott Smiley
Subject: FW: Franklin Editor UI

From: Dikran S Meliksetian [mailto:Dikran_Meliksetian@us.ibm.com]
Sent: Thursday, March 24, 2005 3:33 PM
To: Jon Gibbons
Cc: Louis Weitzman; Sara Elo-Dean
Subject: Fw: Franklin Editor UI

Jon,

Here is another email in in it.
I do not think we will be .

Dikran

---- Forwarded by Dikran S Meliks(

To: John Dorval/Somers/IBM@I
cc: Louis Weitzman/Southbury/IE
From: Sara Elo/Armonk/IBM@IBI
Subject: Re: Franklin Editor UI

hi John, thanks for the feedba
also cc:ing Louis, who develop

Sara

Advanced Internet Technology
http://w3.webahead.ibm.com

To: Sara Elo/Armonk/IBM
cc:
From: John Dorval/Somers/IBM@IBMUS
Subject: Re: Franklin Editor UI

Sara,
Thanks it installed no problem. Your team is doing a terrific job. I'm very impressed with the functionality.
Don't take any of the following comments as criticism -- they are just suggestions for improvements.

- Usually "check out" means to freeze the document while I 'edit' it.
Franklin checkout seems to mean view it (read only).
Actually "Check out" does mean to freeze it for edit. Did you notice that there are two check-out buttons side by side?
The tooltips and the icons themselves can be improved to be singificanlty different.
"Check out document" and "View document"
instead of
"Check out fragment or servable" and "Check out fragment or servable in read only mode"
good point.

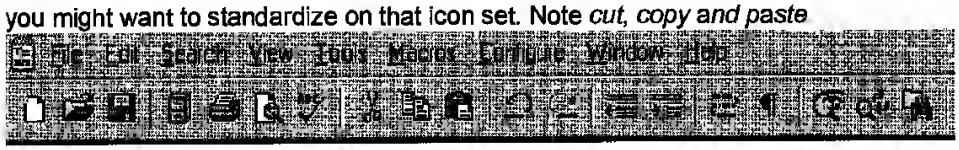
- ISO codes should be stored with the document along with Country name,
language, etc.
The ISO code is actually more important since an appl can look up the name
given the ISO code.
We thought long about this with Patrick Rooney. It seems that it will be better for editors to select from the names not the codes,
as names are more readable. However, if the editor chooses a name, then the code should be filled in automatically by consulting
the infamous centralized taxonomy server, or a local automatically updated copy of it on the Franklin server, when the document is checked in.

In the case of other code+name pairs, such as IBM divisions, again, it's more likely that a name might change but the code
stays the same, so the code should always be the one searched upon and used server side, but the name should be displayed
to the user for viewing or selection.

This is not implemented but is a possible way to do it if we go into pilot

- For items with multiple paragraphs (ie prod description in the Net.commerce entry), need way to mark the paragraphs. The xhtml work that Patrick is doing may help here.
yes, that's the functionality we are currently adding... stay tuned

- There is a paste button on the "Read Only" form. Probably not needed.
good point.
- Got an SAX exception checking out the Netfinity product page
ah, we will check
- Just a general comment, I know it is hard to do but having a wsiwig authoring tool is important. Business uses don't want to see the tags or have to type them. The eSites requirements folks were very strong on that point.
yes, i realize this is becoming the biggest issue here....
- Not sure where some of your button icons come from (maybe they are unix versions which I am unfamilair with). I believe most users will be Windows users so you might want to standardize on that icon set. Note *cut, copy and paste*



- The shades of gray used for the windows us different than the default windows colors and it makes the appl look a little out of place -- again maybe this is a unix color scheme.

Of course, this is all nit picky stuff so I'll stop here.

Thanks.
Regards, John

Enterprise Web Management, Advanced e-Business Technology
Route 100 Somers NY
Tel. (914) 766-1515 TI 826 Fax x-1869
Internet: dorval@us.ibm.com

To: Ron Lautmann/Mountain View/IBM@IBMUS, John Dorval/Somers/IBM@IBMUS, Stephen Kennedy/Somers/IBM@IBMUS, Patrick Rooney/Boulder/IBM@IBMUS
cc: Franklin, Maria Hernandez/Somers/IBM@IBMUS
From: Sara Elo/Armonk/IBM@IBMUS
Subject. Franklin Editor UI

Hi all,

Install:
Please go to http://monolith.adtech.internet.ibm.com/franklin/downloads/index.html for
"How to download Franklin Editor UI" and
"How to get started with Franklin UI"

Once you have the client installed and running you might want to try the following things:

Try out:
1) Search for "Document Type = SoftwareSalesManual", Check-Out from the server the fragment entitled "Net.Commerce". This is based on the DTD Patrick provided, filled in with Net.Commerce data. Feel free to create a new SWSalesManual fragment to see how the authoring works.

2) Search for "Document Type = SWSALESMANUALPAGE" and select the "Net.Commerce" servable you get back. This is a page that imports the fragment in 1) If you merge this document into your Active List, you can preview it to see how a page gets rendered into HTML using LotusXSL. You can also check it out to see its contents.

3) Search for "Document Type = PRODUCTPAGE". Check-Out from the server the "Netfinity" servable and Preview it. This is another example of a Page made up of 4 subfragments rendered in HTML.

4) Go wild, create any fragments and pages to get a feel for the editor. The current server is a play space.

Caveats:
- Please ignore anything entitled "Test" in the search results. Those are our tests...
- In the SWSalesManual, the fields containing <P>, <LI> etc... are not yet rendered correctly in the final HTML. We are working on integrating the mechanism, small matter of programming...

User Names:
Below are your usernames and passwords for the Franklin Editor:

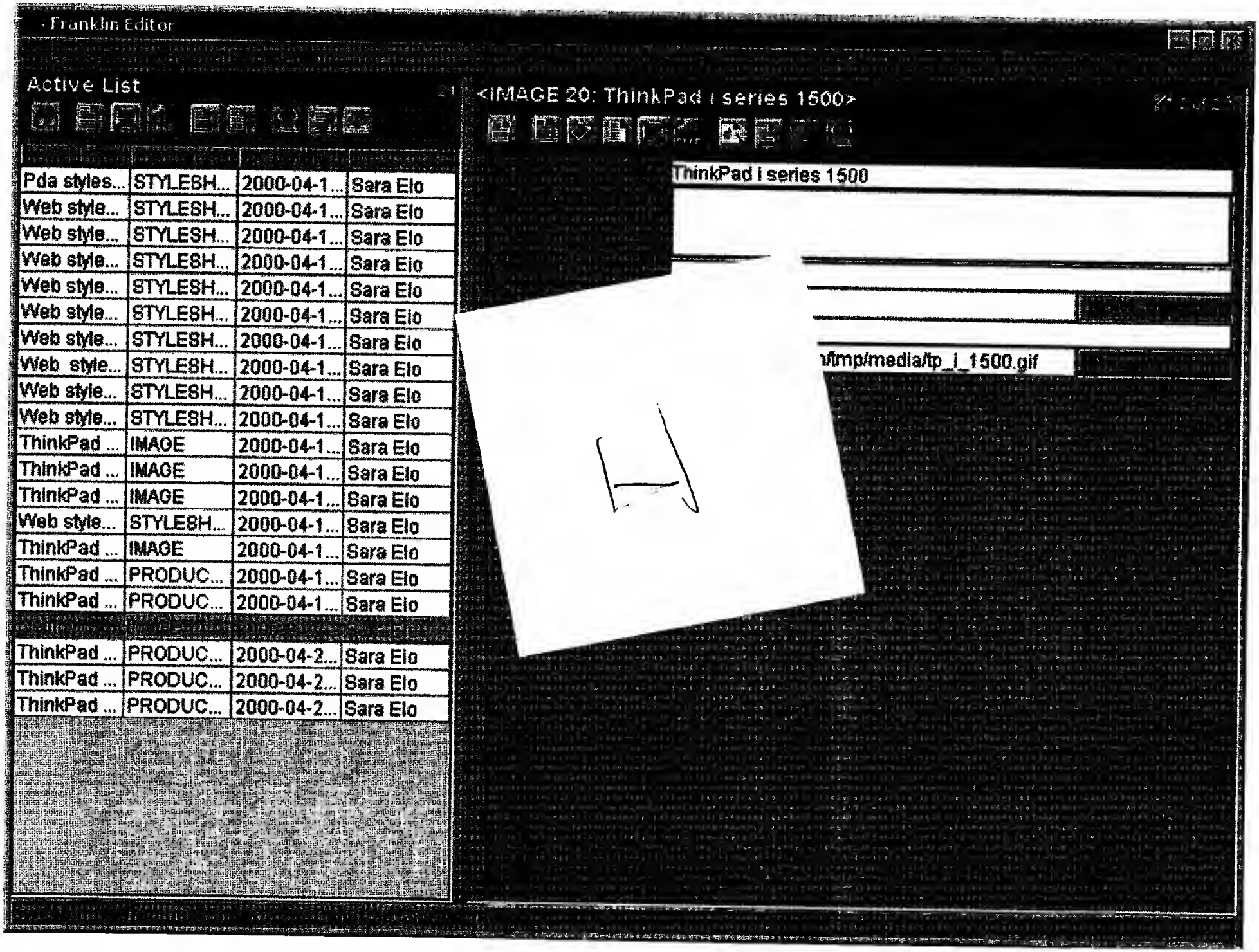
```
<USER>
    <NAME>Ron Lautmann</NAME>
    <EMAIL>lautmann@us.ibm.com</EMAIL>
    <PASSWORD>ron</PASSWORD>
    <ROLE>Editor</ROLE>
```

```
  </USER>
 <USER>
     <NAME>John Dorval</NAME>
     <EMAIL>dorval@us.ibm.com</EMAIL>
     <PASSWORD>john</PASSWORD>
     <ROLE>Editor</ROLE>
  </USER>
 <USER>
     <NAME>Stephen Kennedy</NAME>
     <EMAIL>stephen@us.ibm.com</EMAIL>
     <PASSWORD>stephen</PASSWORD>
     <ROLE>Editor</ROLE>
  </USER>
 <USER>
     <NAME>Patrick Rooney</NAME>
     <EMAIL>rooney@us.ibm.com</EMAIL>
     <PASSWORD>patrick</PASSWORD>
     <ROLE>Editor</ROLE>
  </USER>
</USERS>
```

Don't forget that tooltips exist for all icons....
Comments, bug reports welcome, please send to me and i'll pass on.

Regards,
Sara

.....
Advanced Internet Technology
http://w3.webahead.ibm.com
.....


Franklin Editor
Active List
<IMAGE 20: ThinkPad i series 1500>
ThinkPad i series 1500
/tmp/media/tp_i_1500.gif
Pda styles... STYLESH... 2000-04-1... Sara Elo
Web style... STYLESH... 2000-04-1... Sara Elo
Web style... STYLESH... 2000-04-1... Sara Elo
Web style... STYLESH... 2000-04-1... Sara Elo
Web style... STYLESH... 2000-04-1... Sara Elo
Web style... STYLESH... 2000-04-1... Sara Elo
Web style... STYLESH... 2000-04-1... Sara Elo
Web style... STYLESH... 2000-04-1... Sara Elo
Web style... STYLESH... 2000-04-1... Sara Elo
Web style... STYLESH... 2000-04-1... Sara Elo
ThinkPad ... IMAGE 2000-04-1... Sara Elo
ThinkPad ... IMAGE 2000-04-1... Sara Elo
ThinkPad ... IMAGE 2000-04-1... Sara Elo
Web style... STYLESH... 2000-04-1... Sara Elo
ThinkPad ... IMAGE 2000-04-1... Sara Elo
ThinkPad ... PRODUC... 2000-04-1... Sara Elo
ThinkPad ... PRODUC... 2000-04-1... Sara Elo
ThinkPad ... PRODUC... 2000-04-2... Sara Elo
ThinkPad ... PRODUC... 2000-04-2... Sara Elo
ThinkPad ... PRODUC... 2000-04-2... Sara Elo

Franklin Editor

Active List

<IMAGEFRAGMENT 3: title>

1 of

franklin image test

NONE

/multimedia/images

franklin_small.gif

./tmp/media/franklin_small.gif

Search Interface

Format Query

2000-02-18

Query Results

6087cd0dd1...	IMAGEFRAG...	2000-02-24 0...	title	x	FRAGMENT
6087cd0dd1...	TESTASSOC	2000-02-24 0...	assoc test	x	FRAGMENT
6087cd0dd1...	TEXTFRAGM...	2000-02-23 1...	test	x	FRAGMENT
6087cd0dd1...	LISTFRAGME...	2000-02-22 1...	listfragment	x	FRAGMENT
6087cd0dd1...	LISTFRAGME...	2000-02-22 1...	title	x	FRAGMENT
6087cd0dd1...	TEXTFRAGM...	2000-02-21 2...	xxx	x	FRAGMENT

```
<?xml version="1.0" encoding="US-ASCII" ?>
<!DOCTYPE FRANKLIN_INIT (View Source for full doctype...)>
- <FRANKLIN_INIT>
  - <FRAGMENTS displayname="Fragment">
      <DTD displayname="Text"
        href="http://frasier.dhcp.adtech.internet.ibm.com/franklin/xml/textfragment.dtd
      <DTD displayname="Image"
        href="http://frasier.dhcp.adtech.internet.ibm.com/franklin/xml/imagefragment.d
      <DTD displayname="Audio"
        href="http://frasier.dhcp.adtech.internet.ibm.com/franklin/xml/audiofragment.dt
      <DTD displayname="Video"
        href="http://frasier.dhcp.adtech.internet.ibm.com/franklin/xml/videofragment.dt
    </FRAGMENTS>
  - <SERVABLES displayname="Page">
      <DTD displayname="Page one"
        href="http://frasier.dhcp.adtech.internet.ibm.com/franklin/xml/servableone.dtd"
      <DTD displayname="Page two"
        href="http://frasier.dhcp.adtech.inter                n/franklin/xml/servabletwo.dtd"
      <DTD displayname="Page
        href="http:/                                        l/franklin/xml/foo.dtd" />
    </SERVABLES>
  - <SEARCH>
    - <ATTRIBUTELIS
        <ATTRIBUTE (                                        EATIONTIME"
          class="Time
        <ATTRIBUTE d
          name="LAS1
        <ATTRIBUTE di                                       ENTSIZE"
          class="Integ
        <ATTRIBUTE dis                                      class="Name" />
        <ATTRIBUTE dis                                      ext" />
        <ATTRIBUTE dis                                      s="Text" />
        <ATTRIBUTE disp                                     " class="Text" />
        <ATTRIBUTE disp                                     class="Text" />
        <ATTRIBUTE displ                          TYPE" class="Selection"
          options="http:/          franklin/dtd/entities/types.xml" />
        <ATTRIBUTE displayname="Category" name="CATEGORY" class="Selection"
          options="http://frasier/franklin/dtd/entities/categories.xml" />
      </ATTRIBUTELIST>
    - <CLASSLIST>
      - <CLASS name="Time">
          <OPERATOR>>=</OPERATOR>
          <OPERATOR><=</OPERATOR>
          <OPERATOR>=</OPERATOR>
          <VALUE type="date" />
        </CLASS>
      - <CLASS name="Integer">
          <OPERATOR>>=</OPERATOR>
          <OPERATOR><=</OPERATOR>
          <OPERATOR>=</OPERATOR>
          <VALUE type="Integer" />
        </CLASS>
      - <CLASS name="Name">
```

```
        <OPERATOR>is</OPERATOR>
        <OPERATOR>isn't</OPERATOR>
        <OPERATOR>starts with</OPERATOR>
        <VALUE type="string" />
      </CLASS>
    - <CLASS name="Text">
        <OPERATOR>is</OPERATOR>
        <OPERATOR>starts with</OPERATOR>
        <VALUE type="string" />
      </CLASS>
    - <CLASS name="Selection">
        <OPERATOR>is</OPERATOR>
        <OPERATOR>isn't</OPERATOR>
        <VALUE type="drop-down" />
      </CLASS>
    </CLASSLIST>
  - <RESULTS>
      <ATTRIBUTE displayname="Last Modified Date"
        name="LASTMODIFIEDTIME" class="Time" />
      <ATTRIBUTE displayname="Creator" name="CREATOR" class="Name" />
      <ATTRIBUTE displayname="Title" name="TITLE" class="Text" />
      <ATTRIBUTE displayname="Type" name="TYPE" class="Text" />
    </RESULTS>
  </SEARCH>
</FRANKLIN_INIT>
```

```
1  package com.ibm.adtech.franklin.client;
2
3  import org.w3c.dom.Element;
4  import org.w3c.dom.NodeList;
5  import org.w3c.dom.Document;
6  import org.w3c.dom.Node;
7  import com.ibm.xml.parser.*;
8  import com.ibm.xml.parsers.NonValidatingTXDOMParser;
9  import org.xml.sax.InputSource;
10 import java.awt.event.*;
11 import java.awt.*;
12 import java.net.*;
13 import java.util.*;
14 import java.io.*;
15 import javax.swing.*;
16 import javax.swing.event.*;
17 import sun.misc.BASE64Encoder;
18
19 /**
20  * This type was generated by a SmartGuide.
21  */
22 public class FranklinEditor extends JFrame {
23       // menus
24       private JMenuBar  iv_menuBar          = null;
25       private JMenu     iv_fileMenu         = null;
26       private JMenu     iv_editMenu         = null;
27       private JMenu     iv_viewMenu         = null;
28       private JMenu     iv_helpMenu         = null;
29       private JMenu     iv_pageTypeMenu     = null;
30       private JMenu     iv_fragmentTypeMenu = null;
31       private JMenu     iv_historyMenu      = null;
32
33       // menuItems
34       private JMenuItem iv_exitMenuItem      = null;
35       private JMenuItem iv_backMenuItem      = null;
36       private JMenuItem iv_forwardMenuItem   = null;
37       private JMenuItem iv_toolbarMenuItem   = null;
38       private JMenuItem iv_statusbarMenuItem = null;
39       private JMenuItem iv_aboutBoxMenuItem  = null;
40
41       // buttons
42       private JButton   iv_newButton        = null;
43       private JButton   iv_deleteButton     = null;
44       private JButton   iv_backButton       = null;
45       private JButton   iv_clearButton      = null;
46       private JButton   iv_forwardButton    = null;
47       private JButton   iv_copyButton       = null;
48       private JButton   iv_cutButton        = null;
49       private JButton   iv_pasteButton      = null;
50       private JButton   iv_checkinButton    = null;
51       private JButton   iv_checkoutButton   = null;
52
53       // fonts
54       public static Font cv_smallButtonFont = new Font("Helvetica", Font.BOLD,
55 10);
56       public static Font cv_smallLabelFont  = new Font("Helvetica", Font.BOLD,
57 10);
```

```
58        public static Font cv_labelFont       = new Font("Helvetica", Font.BOLD,
59  12);
60        public static Font cv_buttonFont      = new Font("Helvetica", Font.BOLD,
61  12);
62        public static Font cv_titleFont       = new Font("Helvetica", Font.BOLD,
63  14);
64
65        // colors
66        public static Color cv_requiredInputColor = new Color(255, 128, 128);  //
67  color in ui of required DTD elements
68        public static Color cv_inputColor         = Color.white;                //
69  when filled in return to normal state
70
71        // models
72        private DefaultListModel iv_fragmentTypeModel    = null;          // list
73  of DTD types for fragments
74        private DefaultListModel iv_pageTypeModel        = null;          // list
75  of DTD types for pages
76        private ActiveTableModel iv_activeTableModel     = null;          //
77
78        // flags
79        public static boolean   cv_standaloneP          = true;          // if
80  true, don't go to server for dtd etc
81        public static boolean   cv_debug                = true;          // if
82  true, print out printDebug messages
83
84        // windows & panes
85        public         FragmentManager            iv_fragmentManager            =
86  null;  // manager of current fragments and pages
87        public         InterfaceMaker             iv_interfaceMaker             =
88  null;  // interface maker, has iv for current panel being constructed
89        public  static FranklinEditor             cv_FranklinEditor             =
90  null;
91        public         JPanel                     iv_editorPane                 =
92  null;  // right side content window
93        public         FranklinEditorTypeDialog   iv_typeDialog                 =
94  null;  // dialog for creating fragments or pages
95        private        JPanel         iv_contentPane            = null;
96        //private        JPanel         iv_fragmentPane           = null;
97        //private        JPanel         iv_pagePane               = null;
98        private        JPanel         iv_resultsPane            = null;
99        private        JPanel         iv_fragmentCommandAndData = null;
100       private        JPanel         iv_pageCommandAndData     = null;
101       private        JPanel         iv_resultsCommandAndData  = null;
102       private        JLabel         iv_statusMsg              = null;
103       private        JToolBar       iv_toolBarPane            = null;
104       private        JPanel         iv_statusBarPane          = null;
105       private        JLabel         iv_countLabel             = null;
106       private        JPanel         iv_noPane                 = null;   // when
107 no element is being edited, show this pane
108
109       public  static int            cv_ERROR_MESSAGE          =
110 JOptionPane.ERROR_MESSAGE;
111       public  static int            cv_INFORMATION_MESSAGE    =
112 JOptionPane.INFORMATION_MESSAGE;
113       public  static int            cv_PLAIN_MESSAGE          =
114 JOptionPane.PLAIN_MESSAGE;
```

```
115         public  static int            cv_QUESTION_MESSAGE       =
116   JOptionPane.QUESTION_MESSAGE;
117         public  static int            cv_WARNING_MESSAGE        =
118   JOptionPane.WARNING_MESSAGE;
119
120         // sizes
121         private static int       cv_leftPaneWidth      = 200;
122         private static int       cv_leftPaneMaxWidth   = 500;
123         private static int       cv_leftPaneHeight     = 150;
124         private static int       cv_titleHeight        = 20;
125         public  static int       cv_fragmentLabelWidth = 150;
126         public  static int       cv_fragmentTitleHeight= 15;
127         public  static int       cv_fragmentTextWidth  = 300;
128         public         Dimension iv_screenSize         = null;  // used to center
129   the editor and login dialog box
130
131         // Names
132         public  static String          cv_fragmentDisplayName = "Fragment";
133         public  static String          cv_servableDisplayName = "Page";
134         public  static final String    cv_DATATYPE            = "DATATYPE";  //
135   tag for identifying the UI component
136         public  static final String    cv_CHOICES             = "CHOICES";   //
137   tag for identifying the CHOICES attribute value of DATATYPE
138         public  static final String    cv_SYSTEM_ELEMENT      = "SYSTEM";    //
139   tag elements filled in by system automatically
140         public  static final String    cv_SPECIAL_ELEMENT     = "SPECIAL";   //
141   tag for elements different from universal.dtd
142         public  static final String    cv_PAGETYPE_ELEMENT    = "PAGETYPE";  //
143   tag for PAGETYPE elements
144         public  static final String[]  cv_skipElements        =
145   { cv_SYSTEM_ELEMENT };  // element tags of dtd to skip when building ui
146         public  static final String[]  cv_nonDisplayElements  =
147   { cv_SPECIAL_ELEMENT,   // element tags of dtd not to display (but do kids)
148   when building ui
149
150                 cv_PAGETYPE_ELEMENT};
151         public  TXDocument iv_initDocument = null;    // document object that is
152   the result of loading franklin_init.xml
153         private String     iv_sessionID    = null;    // session ID received from
154   the logging in process.
155   /**
156    * FranklinEditor constructor comment.
157    */
158   public FranklinEditor() {
159         super();
160   }
161   /**
162    * Insert the method's description here.
163    * Creation date: (10/7/99 7:13:39 PM)
164    */
165   public void back() {
166         iv_fragmentManager.back();
167         redisplayEditorPane();
168   }
169   /**
170    * Checkin setups up the document in the fragment (updating it from the ui)
171    * validates it, and sends it to the server. If all works, it updates the
```

```
172    * SYSTEM data in the document object from the result TXDocument returned.
173    * ++++ if checkin works, update the fragments uid. and last modified time, etc
174    *
175    * Creation date: (10/8/99 11:15:52 AM)
176    */
177   public void checkin() {
178         ClientFragment lv_fragment = iv_fragmentManager.getCurrentFragment();
179   // get the current fragment/page being edited
180         if (lv_fragment == null) {
181               FranklinEditor.printDebug("No fragment or page to checkin");
182               setStatusMessage("Nothing to check in");
183         }
184         else {
185               try {
186                     FranklinEditor.printDebug("Checking in " + lv_fragment);
187                     setStatusMessage("Checking in " + lv_fragment);
188                     lv_fragment.checkinSetup();
189   // grab content out of ui,
190                     boolean lv_isValid = lv_fragment.validate();
191   // also highlights ui errors
192                     if (lv_isValid == true) {
193                           lv_fragment.iv_document.printWithFormat(new
194   PrintWriter(System.out));
195                           TXDocument lv_checkinResults =
196   Dispatcher.checkin(lv_fragment);                  // get results back from
197   dispatcher (on server)
198                           if (cv_debug) {
199                                 lv_checkinResults.printWithFormat(new
200   PrintWriter(System.out));              // print result document to stream
201                           }
202                           removeFragmentOrPage(lv_fragment);
203   // remove from the ui
204                           lv_fragment.replaceSystemTag(lv_checkinResults);
205   // add SYSTEM tags back into the fragment
206                           lv_fragment.checkinCacheActiveHeaders();
207                           iv_activeTableModel.addFragment(lv_fragment);
208   // add to active list
209                     }
210                     else {
211                           setStatusMessage("Error validating " + lv_fragment);
212                           showMessageDialog("Document is invalid. Please edit
213   highlighted elements and resubmit",
214
215   FranklinEditor.cv_ERROR_MESSAGE);
216                     }
217               }
218               catch (IOException lv_e) {
219                     System.out.println("Error printing checkin results set");
220               }
221         }
222   }
223   /**
224    * checkout the currently selected set of fragments or pages
225    * if one just do it. if more than one confirm selections, or use progress bar
226    * that the user can cancel after a certain number have been checked out
227    */
228   public void checkout() {
```

```
229         FranklinEditor.printDebug("checkout");
230   }
231   /**
232    * Clear the interface widgets for the current fragment or page
233    */
234   public void clearCurrentFragmentOrPage() {
235         System.out.println("clearCurrentFragmentOrPage");
236         ClientFragment lv_fragment = iv_fragmentManager.getCurrentFragment();
237         if (lv_fragment != null) {
238               lv_fragment.clearInterface();
239         }
240   }
241   /**
242    * Insert the method's description here.
243    * Creation date: (9/29/99 11:36:37 AM)
244    */
245   public void clearStatusMessage() {
246         iv_statusMsg.setText("");
247   }
248   /**
249    * Create a new fragment object of type lv_type
250    *   if not in DTD hashtable for fragments read in the DTD,
251    *         save DTD in hashtable
252    *   create an instance of this DTD as a ClientFragment object (which extends
253   TXDocument)
254    *   add to history list (history list(vector), current item)
255    *   display ClientFragment (save UI widgets on ClientFragment)
256    *
257    * Creation date: (10/7/99 2:24:27 PM)
258    * @param lv_type java.lang.String
259    */
260   public void createFragment(String lv_name) {
261         printDebug("createFragment = " + lv_name);
262
263         DTD lv_dtd = Dispatcher.getFragmentDTD(lv_name);
264         if (lv_dtd == null) {
265               System.out.println("FranklinEditor.createFragment: Error loading '"
266   + lv_name + "' DTD from server");
267               return;
268         }
269         createFragmentInternal(lv_dtd);
270   }
271   /**
272    * Common tasks for creating a fragment or page ClientFragment object
273    *   create an instance of this DTD as a ClientFragment object (which extends
274   TXDocument)
275    *   add to history list (history list(vector), current item)
276    *   attach method to go to fragment from history menu
277    *   display ClientFragment (save UI widgets on ClientFragment)
278    *
279    * Creation date: (10/7/99 2:24:27 PM)
280    * @param lv_type java.lang.String
281    */
282   public void createFragmentInternal(DTD lv_dtd) {
283         // create client fragment
284         ClientFragment lv_fragment = new ClientFragment(iv_fragmentManager,
285   lv_dtd);
```

```
286
287         // add to history menu
288         iv_historyMenu.add(lv_fragment.getMenuItem());
289
290         // redisplay the editor pane
291         redisplayEditorPane();
292   }
293   /**
294    * create a new page object of type lv_type.
295    * add to history list
296    * and DTD to dtd hashtable
297    * Creation date: (10/7/99 2:24:27 PM)
298    * @param lv_type java.lang.String
299    */
300   public void createPage(String lv_name) {
301         printDebug("createPage = " + lv_name);
302
303         DTD lv_dtd = Dispatcher.getPageDTD(lv_name);
304         if (lv_dtd == null) {
305               System.out.println("FranklinEditor.createPage: Error loading '" +
306   lv_name + "' DTD from server");
307               return;
308         }
309         createFragmentInternal(lv_dtd);
310   }
311   /**
312    * delete fragement or page from UI and backend server
313    * Creation date: (9/29/99 11:36:37 AM)
314    */
315   public void deleteCurrentFragmentOrPage() {
316         System.out.println("deleteCurrentFragmentOrPage");
317         ClientFragment lv_fragment = iv_fragmentManager.getCurrentFragment();
318         if (lv_fragment != null) {
319               removeCurrentFragmentOrPage();
320   // remove from ui
321               String lv_status = Dispatcher.deleteFragment(lv_fragment);
322   // delete from server
323               setStatusMessage(lv_status);
324         }
325   }
326   /**
327    * Insert the method's description here.
328    * Creation date: (10/7/99 7:13:39 PM)
329    */
330   public void editCopy() {
331         setStatusMessage("Copy not yet implemented");
332   }
333   /**
334    * Insert the method's description here.
335    * Creation date: (10/7/99 7:13:39 PM)
336    */
337   public void editCut() {
338         setStatusMessage("Cut not yet implemented");
339
340         /*
341         // this is example of how to create xml from TXDocument
342         try {
```

```
343               StringWriter lv_writer = new StringWriter();
344               iv_initDocument.toXMLString(lv_writer);
345               System.out.println("xml from iv_initDocument -----");
346               //System.out.println(lv_writer.toString());
347
348               com.ibm.adtech.franklin.Fragment lv_fragment =
349   com.ibm.adtech.franklin.Fragment.XML2Fragment(lv_writer.toString());
350               System.out.println(lv_fragment);
351               System.out.println(lv_fragment.toXMLString());
352
353
354         }
355         catch (IOException lv_e) {
356               System.out.println("editCut: Error trying to write XML for
357   iv_initDocument");
358         }
359         */
360   }
361   /**
362    * Insert the method's description here.
363    * Creation date: (10/7/99 7:13:39 PM)
364    */
365   public void editPaste() {
366         setStatusMessage("Paste not yet implemented");
367   }
368   /**
369    * exitFranklinEditor:
370   (ExitMenuItem.action.actionPerformed(java.awt.event.ActionEvent) -->
371   FranklinEditor.dispose()V)
372    * @param arg1 java.awt.event.ActionEvent
373    */
374   private void exitFranklinEditor() {
375         try {
376               if (!cv_standaloneP) {
377                     Dispatcher.logout(this);
378               }
379               this.dispose();
380         }
381         catch (java.lang.Throwable ivjExc) {
382               handleException(ivjExc);
383         }
384   }
385   /**
386    * Insert the method's description here.
387    * Creation date: (9/29/99 11:36:37 AM)
388    */
389   public void expandResultsPane() {
390         System.out.println("expandResultsPane");
391         Dimension lv_dim = iv_resultsPane.getSize();
392         lv_dim.height = 0;
393         iv_resultsCommandAndData.setMinimumSize(lv_dim);
394         lv_dim.height = 100;
395         iv_resultsCommandAndData.setPreferredSize(lv_dim);
396         lv_dim.height = 500;
397         iv_resultsCommandAndData.setMaximumSize(lv_dim);
398         iv_resultsPane.doLayout();
399   }
```

```
400   /**
401    * Insert the method's description here.
402    * Creation date: (10/7/99 7:13:39 PM)
403    */
404   public void forward() {
405         iv_fragmentManager.forward();
406         redisplayEditorPane();
407   }
408   /**
409    * Insert the method's description here.
410    * Creation date: (9/29/99 11:36:37 AM)
411    */
412   public JMenuItem getAboutBoxMenuItem() {
413         return iv_aboutBoxMenuItem;
414   }
415   /**
416    * Insert the method's description here.
417    * Creation date: (9/29/99 11:36:37 AM)
418    */
419   private JPanel getActiveTablePane() {
420         JPanel lv_resultsPane     = new JPanel();
421         JPanel lv_title           = new JPanel();
422         JPanel lv_commands        = new JPanel();
423         JLabel lv_titleLabel      = new JLabel(" Active List",
424   SwingConstants.LEFT);
425         iv_countLabel             = new JLabel("0 ",
426   SwingConstants.RIGHT);
427         JButton lv_searchButton   = new JButton("Search");
428         JButton lv_viewButton     = new JButton("View");
429         JButton lv_checkoutButton = new JButton("Checkout");
430         iv_resultsCommandAndData  = new JPanel();
431
432         // fonts
433         lv_searchButton.setFont(cv_buttonFont);
434         lv_viewButton.setFont(cv_buttonFont);
435         lv_checkoutButton.setFont(cv_buttonFont);
436         lv_titleLabel.setFont(cv_titleFont);
437         iv_countLabel.setFont(cv_smallLabelFont);
438
439         // colors
440         iv_resultsCommandAndData.setBackground(Color.white);
441         lv_title.setBackground(Color.black);
442         lv_titleLabel.setForeground(Color.white);
443         iv_countLabel.setForeground(Color.white);
444
445         iv_activeTableModel   = new ActiveTableModel(this);
446         JTable lv_activeTable = new JTable(iv_activeTableModel);
447         JScrollPane lv_scroll = new JScrollPane(lv_activeTable);
448
449         lv_title.setLayout(new BorderLayout());
450         lv_title.add(lv_titleLabel, "West");
451         lv_title.add(iv_countLabel, "East");
452
453         lv_commands.setLayout(new BoxLayout(lv_commands, BoxLayout.X_AXIS));
454         lv_commands.add(lv_searchButton);
455         lv_commands.add(lv_viewButton);
456         lv_commands.add(lv_checkoutButton);
```

```
457         lv_commands.add(Box.createHorizontalGlue());
458
459         iv_resultsCommandAndData.setLayout(new BorderLayout());
460         iv_resultsCommandAndData.add(lv_commands, "North");
461         iv_resultsCommandAndData.add(lv_scroll,   "Center");
462
463         lv_resultsPane.setLayout(new BorderLayout());
464         lv_resultsPane.add(lv_title,                  "North");
465         lv_resultsPane.add(iv_resultsCommandAndData,  "Center");
466
467         lv_resultsPane.setPreferredSize(new
468   Dimension(cv_leftPaneWidth,cv_leftPaneHeight));
469
470         return lv_resultsPane;
471
472
473         // ToolTips
474         //lv_toggleButton.setToolTipText("Collapse active list pane");
475
476         /**
477         lv_toggleButton.addItemListener(new ItemListener() {
478               public void itemStateChanged(ItemEvent lv_e) {
479                     JToggleButton lv_b = (JToggleButton)lv_e.getSource();
480                     if (lv_b.getModel().isSelected()) {
481                           setStatusMessage("Collapsing active list pane");
482                           lv_b.setToolTipText("Expand active list pane");
483                           lv_b.setText("Open");
484                           collapseResultsPane();
485                     }
486                     else {
487                           setStatusMessage("Expanding active list pane");
488                           lv_b.setToolTipText("Collapse active list pane");
489                           lv_b.setText("Close");
490                           expandResultsPane();
491                     }
492               }
493         });
494
495         lv_toggleButton.addChangeListener(new ChangeListener() {
496               public void stateChanged(ChangeEvent lv_e) {
497                     //System.out.println("change listener");
498               }
499         });
500
501         lv_toggleButton.addActionListener(new ActionListener() {
502               public void actionPerformed(ActionEvent lv_e) {
503                     //System.out.println("action event");
504               }
505         });
506         */
507
508
509   }
510   /**
511    * Insert the method's description here.
512    * Creation date: (10/8/99 10:54:36 AM)
513    * @return javax.swing.JMenuItem
```

```
514    */
515   public JMenuItem getBackMenuItem() {
516         return iv_backMenuItem;
517   }
518   /**
519    * Return the EditMenu property value.
520    * @return javax.swing.JMenu
521    */
522   private javax.swing.JMenu getEditMenu() {
523         if (iv_editMenu == null) {
524               try {
525
526                     JMenuItem lv_copyMenuItem = new javax.swing.JMenuItem();
527                     lv_copyMenuItem.setName("CopyMenuItem");
528                     lv_copyMenuItem.setText("Copy");
529
530                     JMenuItem lv_undoMenuItem = new javax.swing.JMenuItem();
531                     lv_undoMenuItem.setName("UndoMenuItem");
532                     lv_undoMenuItem.setText("Undo");
533
534                     JMenuItem lv_redoMenuItem = new javax.swing.JMenuItem();
535                     lv_redoMenuItem.setName("RedoMenuItem");
536                     lv_redoMenuItem.setText("Redo");
537
538                     JMenuItem lv_cutMenuItem = new javax.swing.JMenuItem();
539                     lv_cutMenuItem.setName("CutMenuItem");
540                     lv_cutMenuItem.setText("Cut");
541
542                     JMenuItem lv_pasteMenuItem = new javax.swing.JMenuItem();
543                     lv_pasteMenuItem.setName("PasteMenuItem");
544                     lv_pasteMenuItem.setText("Paste");
545
546                     JMenuItem lv_removeMenuItem = new javax.swing.JMenuItem();
547                     lv_removeMenuItem.setName("RemoveMenuItem");
548                     lv_removeMenuItem.setText("Remove");
549                     lv_removeMenuItem.addActionListener(new ActionListener() {
550                           public void actionPerformed(ActionEvent lv_e) {
551                                 removeCurrentFragmentOrPage();
552                           }
553                     });
554
555                     JMenuItem lv_deleteMenuItem = new javax.swing.JMenuItem();
556                     lv_deleteMenuItem.setName("DeleteMenuItem");
557                     lv_deleteMenuItem.setText("Delete");
558                     lv_deleteMenuItem.addActionListener(new ActionListener() {
559                           public void actionPerformed(ActionEvent lv_e) {
560                                 deleteCurrentFragmentOrPage();
561                           }
562                     });
563
564                     JMenuItem lv_selectAllMenuItem = new javax.swing.JMenuItem();
565                     lv_selectAllMenuItem.setName("Select_AllMenuItem");
566                     lv_selectAllMenuItem.setText("Select All");
567
568                     JMenuItem lv_findReplaceMenuItem = new
569   javax.swing.JMenuItem();
570                     lv_findReplaceMenuItem.setName("Find_ReplaceMenuItem");
```

```
571                     lv_findReplaceMenuItem.setText("Find/Replace");
572
573                     iv_editMenu = new javax.swing.JMenu();
574                     iv_editMenu.setName("EditMenu");
575                     iv_editMenu.setText("Edit");
576
577
578                     iv_editMenu.add(lv_undoMenuItem);
579                     iv_editMenu.add(lv_redoMenuItem);
580                     iv_editMenu.add(new JSeparator());
581                     iv_editMenu.add(lv_cutMenuItem);
582                     iv_editMenu.add(lv_copyMenuItem);
583                     iv_editMenu.add(lv_pasteMenuItem);
584                     iv_editMenu.add(new JSeparator());
585                     iv_editMenu.add(lv_removeMenuItem);
586                     iv_editMenu.add(lv_deleteMenuItem);
587                     iv_editMenu.add(lv_selectAllMenuItem);
588                     iv_editMenu.add(lv_findReplaceMenuItem);
589
590               }
591               catch (java.lang.Throwable ivjExc) {
592                     handleException(ivjExc);
593               }
594         }
595         return iv_editMenu;
596   }
597   /**
598    * Insert the method's description here.
599    * Creation date: (9/29/99 11:36:37 AM)
600    */
601   public JMenuItem getExitMenuItem() {
602         return iv_exitMenuItem;
603   }
604   /**
605    * Return the FileMenu property value.
606    * @return javax.swing.JMenu
607    */
608   private javax.swing.JMenu getFileMenu() {
609         if (iv_fileMenu == null) {
610               try {
611                     JMenuItem lv_searchMenuItem = new javax.swing.JMenuItem();
612                     lv_searchMenuItem.setName("SearchMenuItem");
613                     lv_searchMenuItem.setText("Search");
614                     //lv_searchMenuItem.setActionCommand("OpenFragment");
615                     lv_searchMenuItem.addActionListener(new ActionListener() {
616                           public void actionPerformed(ActionEvent lv_e) {
617                                 search();
618                           }
619                     });
620
621                     JMenuItem lv_saveMenuItem = new javax.swing.JMenuItem();
622                     lv_saveMenuItem.setName("SaveMenuItem");
623                     lv_saveMenuItem.setText("Check In");
624                     lv_saveMenuItem.addActionListener(new ActionListener() {
625                           public void actionPerformed(ActionEvent lv_e) {
626                                 checkin();
627                           }
```

```
628                     });
629
630                     JMenuItem lv_loadAllMenuItem = new javax.swing.JMenuItem();
631                     lv_loadAllMenuItem.setName("LoadAllMenuItem");
632                     lv_loadAllMenuItem.setText("Load All DTDs");
633                     lv_loadAllMenuItem.addActionListener(new ActionListener() {
634                           public void actionPerformed(ActionEvent lv_e) {
635                                 loadAllDTDs();
636                           }
637                     });
638
639                     JMenuItem lv_reloadMenuItem = new javax.swing.JMenuItem();
640                     lv_reloadMenuItem.setName("ReloadMenuItem");
641                     lv_reloadMenuItem.setText("Initialize DTDs");
642                     lv_reloadMenuItem.addActionListener(new ActionListener() {
643                           public void actionPerformed(ActionEvent lv_e) {
644                                 reloadDTDs();
645                           }
646                     });
647
648                     iv_exitMenuItem = new javax.swing.JMenuItem();
649                     iv_exitMenuItem.setName("ExitMenuItem");
650                     iv_exitMenuItem.setText("Exit");
651                     iv_exitMenuItem.addActionListener(new ActionListener() {
652                           public void actionPerformed(ActionEvent lv_e) {
653                                 exitFranklinEditor();
654                           }
655                     });
656
657                     iv_fileMenu = new javax.swing.JMenu();
658                     iv_fileMenu.setName("FileMenu");
659                     iv_fileMenu.setText("File");
660                     iv_fileMenu.add(getFragmentTypeMenu());
661                     iv_fileMenu.add(getPageTypeMenu());
662                     iv_fileMenu.add(new JSeparator());
663                     iv_fileMenu.add(lv_loadAllMenuItem);
664                     iv_fileMenu.add(lv_reloadMenuItem);
665                     iv_fileMenu.add(new JSeparator());
666                     iv_fileMenu.add(lv_searchMenuItem);
667                     iv_fileMenu.add(new JSeparator());
668                     iv_fileMenu.add(lv_saveMenuItem);
669                     iv_fileMenu.add(new JSeparator());
670                     iv_fileMenu.add(iv_exitMenuItem);
671
672               } catch (java.lang.Throwable ivjExc) {
673                     handleException(ivjExc);
674               }
675         }
676         return iv_fileMenu;
677   }
678   /**
679    * Insert the method's description here.
680    * Creation date: (10/8/99 10:54:36 AM)
681    * @return javax.swing.JMenuItem
682    */
683   public JMenuItem getForwardMenuItem() {
684         return iv_forwardMenuItem;
```

```
}
/**
 * Insert the method's description here.
 * Creation date: (9/29/99 11:36:37 AM)
 */
private JPanel getFragmentPane() {
      JPanel lv_fragmentPane        = new JPanel();
      JPanel lv_title               = new JPanel();
      JPanel lv_commands            = new JPanel();
      JLabel lv_titleLabel          = new JLabel(" Fragments",
SwingConstants.LEFT);
      JToggleButton lv_toggleButton = new JToggleButton("Close");
      JButton       lv_createButton = new JButton("Create");
      iv_fragmentCommandAndData     = new JPanel();

      // Fonts
      lv_createButton.setFont(cv_buttonFont);
      lv_titleLabel.setFont(cv_titleFont);
      lv_toggleButton.setFont(cv_smallButtonFont);
      // Colors
      lv_title.setBackground(Color.black);
      lv_titleLabel.setForeground(Color.white);
      // ToolTips
      lv_toggleButton.setToolTipText("Collapse fragment pane");

      lv_toggleButton.addItemListener(new ItemListener() {
            public void itemStateChanged(ItemEvent lv_e) {
                  JToggleButton lv_b = (JToggleButton)lv_e.getSource();
                  if (lv_b.getModel().isSelected()) {
                        setStatusMessage("Collapsing fragment pane");
                        lv_b.setToolTipText("Expand fragment pane");
                        lv_b.setText("Open");
                        //collapseFragmentPane();
                  }
                  else {
                        setStatusMessage("Expanding fragment pane");
                        lv_b.setToolTipText("Collapse fragment pane");
                        lv_b.setText("Close");
                        //expandFragmentPane();
                  }
            }
      });

      JList lv_fragmentList = new JList(getFragmentTypeModel());
      JScrollPane lv_scroll = new JScrollPane(lv_fragmentList);

      lv_title.setLayout(new BorderLayout());
      lv_title.add(lv_titleLabel,   "West");
      lv_title.add(lv_toggleButton, "East");

      lv_commands.setLayout(new BoxLayout(lv_commands, BoxLayout.X_AXIS));
      lv_commands.add(lv_createButton);
      lv_commands.add(Box.createHorizontalGlue());

      iv_fragmentCommandAndData.setLayout(new BorderLayout());
      iv_fragmentCommandAndData.add(lv_commands, "North");
      iv_fragmentCommandAndData.add(lv_scroll,   "Center");
```

```
742
743         lv_fragmentPane.setLayout(new BorderLayout());
744         lv_fragmentPane.add(lv_title,                   "North");
745         lv_fragmentPane.add(iv_fragmentCommandAndData, "Center");
746
747         lv_fragmentPane.setBounds(0,0,cv_leftPaneWidth,cv_leftPaneHeight);
748
749
750         return lv_fragmentPane;
751   }
752   /**
753    * return the fragmentTypeMenu. the models should have been initialized by now
754    * so we can go to them to pick out the names to populate the menu.
755    * also add action so that when we click on the fragment, it will create an
756   instance
757    * @return javax.swing.JMenu
758    */
759
760   private javax.swing.JMenu getFragmentTypeMenu() {
761         if (iv_fragmentTypeMenu == null) {
762               try {
763                     iv_fragmentTypeMenu = new javax.swing.JMenu();
764                     iv_fragmentTypeMenu.setName("FragmentTypeMenu");
765                     iv_fragmentTypeMenu.setText("New " + cv_fragmentDisplayName);
766                     //iv_fragmentTypeMenu.setActionCommand("NewFragment");
767
768                     DefaultListModel lv_model = getFragmentTypeModel();
769                     for (int i = 0; i < lv_model.size(); i++) {
770                           String lv_name = (String)lv_model.elementAt(i);
771                           JMenuItem lv_menuItem = new JMenuItem();
772                           lv_menuItem.setName("MenuItem" + (i + 1));
773                           lv_menuItem.setText(lv_name);
774                           iv_fragmentTypeMenu.add(lv_menuItem);
775                           lv_menuItem.addActionListener(new ActionListener() {
776                                 public void actionPerformed(ActionEvent lv_e) {
777
778         createFragment(((JMenuItem)(lv_e.getSource())).getText());
779                                 }
780                           });
781                     }
782
783               } catch (java.lang.Throwable ivjExc) {
784                     handleException(ivjExc);
785               }
786         }
787         return iv_fragmentTypeMenu;
788   }
789   /**
790    * Insert the method's description here.
791    * Creation date: (9/29/99 11:36:37 AM)
792    */
793   public DefaultListModel getFragmentTypeModel() {
794         if (iv_fragmentTypeModel == null) {
795               iv_fragmentTypeModel = new DefaultListModel();
796         }
797         return iv_fragmentTypeModel;
798   }
```

```
799   /**
800    * Return the FranklinEditorJMenuBar property value.
801    * @return javax.swing.JMenuBar
802    */
803   /* WARNING: THIS METHOD WILL BE REGENERATED. */
804   private javax.swing.JMenuBar getFranklinEditorJMenuBar() {
805         if (iv_menuBar == null) {
806               try {
807                     iv_menuBar = new javax.swing.JMenuBar();
808                     iv_menuBar.setName("FranklinEditorMenuBar");
809                     iv_menuBar.add(getFileMenu());
810                     iv_menuBar.add(getEditMenu());
811                     iv_menuBar.add(getViewMenu());
812                     iv_menuBar.add(getHelpMenu());
813
814               }
815               catch (java.lang.Throwable ivjExc) {
816                     handleException(ivjExc);
817               }
818         }
819         return iv_menuBar;
820   }
821   /**
822    * Return the HelpMenu property value.
823    * @return javax.swing.JMenu
824    */
825   /* WARNING: THIS METHOD WILL BE REGENERATED. */
826   private javax.swing.JMenu getHelpMenu() {
827         if (iv_helpMenu == null) {
828               try {
829                     iv_aboutBoxMenuItem = new javax.swing.JMenuItem();
830                     iv_aboutBoxMenuItem.setName("AboutBoxMenuItem");
831                     iv_aboutBoxMenuItem.setText("About Franklin");
832
833                     iv_aboutBoxMenuItem.addActionListener(new ActionListener() {
834                           public void actionPerformed(ActionEvent lv_e) {
835                                 showAboutBox();
836                           }
837                     });
838
839                     JMenuItem lv_helpTopicsMenuItem = new
840   javax.swing.JMenuItem();
841                     lv_helpTopicsMenuItem.setName("Help_TopicsMenuItem");
842                     lv_helpTopicsMenuItem.setText("Help Topics");
843
844                     lv_helpTopicsMenuItem.addActionListener(new ActionListener()
845   {
846                           public void actionPerformed(ActionEvent lv_e) {
847                                 showHelpTopics();
848                           }
849                     });
850
851                     iv_helpMenu = new javax.swing.JMenu();
852                     iv_helpMenu.setName("HelpMenu");
853                     iv_helpMenu.setText("Help");
854
855                     iv_helpMenu.add(lv_helpTopicsMenuItem);
```

```
856                     iv_helpMenu.add(iv_aboutBoxMenuItem);
857               }
858               catch (java.lang.Throwable ivjExc) {
859                     handleException(ivjExc);
860               }
861         }
862         return iv_helpMenu;
863   }
864   /**
865    * Return the JFrameContentPane property value.
866    * @return javax.swing.JPanel
867    */
868   private javax.swing.JPanel getJFrameContentPane() {
869         if (iv_contentPane == null) {
870               try {
871                     // get 3 left column panes
872                     //iv_fragmentPane     = getFragmentPane();
873                     //iv_pagePane         = getPagePane();
874                     iv_resultsPane      = getActiveTablePane();
875                     JPanel lv_extraPane = new JPanel();              /// extra
876   pane when everything is collapsed
877                     JScrollPane lv_scrollPane = null;
878
879                     // right side editor pane
880                     iv_editorPane = getNoPane();
881                     lv_scrollPane = new JScrollPane(iv_editorPane);
882
883                     // horizontal split pane
884                     JSplitPane lv_horizontalSplitPane = new
885   javax.swing.JSplitPane(javax.swing.JSplitPane.HORIZONTAL_SPLIT);
886                     lv_horizontalSplitPane.setName("HorizontalSplitPane");
887                     lv_horizontalSplitPane.add(iv_resultsPane, "left");
888                     lv_horizontalSplitPane.add(lv_scrollPane,  "right");
889
890                     iv_contentPane = new javax.swing.JPanel();
891                     iv_contentPane.setName("JFrameContentPane");
892                     iv_contentPane.setLayout(new java.awt.BorderLayout());
893                     iv_contentPane.add(getToolBarPane(),       "North");
894                     iv_contentPane.add(lv_horizontalSplitPane, "Center");
895                     iv_contentPane.add(getStatusBarPane(),     "South");
896
897
898               } catch (java.lang.Throwable ivjExc) {
899                     handleException(ivjExc);
900               }
901         }
902         return iv_contentPane;
903   }
904   /**
905    * Insert the method's description here.
906    * Creation date: (9/29/99 11:36:37 AM)
907    */
908   public JButton getNewToolbarButton() {
909         return iv_newButton;
910   }
911   /**
912    * return the iv_noPane panel for display when no element is being edited
```

```
913    * Creation date: (10/12/99 8:55:10 AM)
914    */
915   private JPanel getNoPane() {
916         if (iv_noPane == null) {
917               iv_noPane = new JPanel();
918               JLabel lv_label = new JLabel("No fragment being edited",
919   SwingConstants.LEFT);
920               lv_label.setFont(cv_titleFont);
921               iv_noPane.setLayout(new BorderLayout());
922               iv_noPane.add(lv_label, "North");
923         }
924         return iv_noPane;
925   }
926   /**
927    * Insert the method's description here.
928    * Creation date: (9/29/99 11:36:37 AM)
929    */
930   private JPanel getPagePane() {
931         JPanel lv_pagePane     = new JPanel();
932         JPanel lv_title        = new JPanel();
933         JPanel lv_commands     = new JPanel();
934         JLabel lv_titleLabel   = new JLabel(" Pages", SwingConstants.LEFT);
935         JToggleButton lv_toggleButton = new JToggleButton("Close");
936         JButton       lv_createButton = new JButton("Create");
937         iv_pageCommandAndData  = new JPanel();
938
939         // Fonts
940         lv_createButton.setFont(cv_buttonFont);
941         lv_titleLabel.setFont(cv_titleFont);
942         lv_toggleButton.setFont(cv_smallButtonFont);
943         // Colors
944         lv_title.setBackground(Color.black);
945         lv_titleLabel.setForeground(Color.white);
946         // ToolTips
947         lv_toggleButton.setToolTipText("Collapse page pane");
948
949         lv_toggleButton.addItemListener(new ItemListener() {
950               public void itemStateChanged(ItemEvent lv_e) {
951                     JToggleButton lv_b = (JToggleButton)lv_e.getSource();
952                     if (lv_b.getModel().isSelected())
953   {                         // expand
954                           setStatusMessage("Collapsing page pane");
955                           lv_b.setToolTipText("Expand page pane");
956                           lv_b.setText("Open");
957                           //collapsePagePane();
958                     }
959                     else
960   {                                                   // collapse
961                           setStatusMessage("Expanding page pane");
962                           lv_b.setToolTipText("Collapse page pane");
963                           lv_b.setText("Close");
964                           //expandPagePane();
965                     }
966               }
967         });
968
969         lv_toggleButton.addChangeListener(new ChangeListener() {
```

```
970               public void stateChanged(ChangeEvent lv_e) {
971                     //System.out.println("change listener");
972               }
973         });
974
975         lv_toggleButton.addActionListener(new ActionListener() {
976               public void actionPerformed(ActionEvent lv_e) {
977                     //System.out.println("action event");
978               }
979         });
980
981         JList lv_pageList     = new JList(getPageTypeModel());
982         JScrollPane lv_scroll = new JScrollPane(lv_pageList);
983
984         lv_title.setLayout(new BorderLayout());
985         lv_title.add(lv_titleLabel,   "West");
986         lv_title.add(lv_toggleButton, "East");
987
988         lv_commands.setLayout(new BoxLayout(lv_commands, BoxLayout.X_AXIS));
989         lv_commands.add(lv_createButton);
990         lv_commands.add(Box.createHorizontalGlue());
991
992         iv_pageCommandAndData.setLayout(new BorderLayout());
993         iv_pageCommandAndData.add(lv_commands, "North");
994         iv_pageCommandAndData.add(lv_scroll,   "Center");
995
996         lv_pagePane.setLayout(new BorderLayout());
997         lv_pagePane.add(lv_title,              "North");
998         lv_pagePane.add(iv_pageCommandAndData, "Center");
999
1000        lv_pagePane.setBounds(0,0,cv_leftPaneWidth,cv_leftPaneHeight);
1001
1002        return lv_pagePane;
1003  }
1004  /**
1005   * Return the iv_pageTypeMenu. the list models should have been initialized
1006   * by now so we can use them to create the list of possible servable pages
1007   * @return javax.swing.JMenu
1008   */
1009  private javax.swing.JMenu getPageTypeMenu() {
1010        if (iv_pageTypeMenu == null) {
1011              try {
1012                    iv_pageTypeMenu = new javax.swing.JMenu();
1013                    iv_pageTypeMenu.setName("PageTypeMenu");
1014                    iv_pageTypeMenu.setText("New " + cv_servableDisplayName);
1015
1016                    DefaultListModel lv_model = getPageTypeModel();
1017                    for (int i = 0; i < lv_model.size(); i++) {
1018                          String lv_name = (String)lv_model.elementAt(i);
1019                          JMenuItem lv_menuItem = new JMenuItem();
1020                          lv_menuItem.setName("MenuItem" + (i + 1));
1021                          lv_menuItem.setText(lv_name);
1022                          iv_pageTypeMenu.add(lv_menuItem);
1023                          lv_menuItem.addActionListener(new ActionListener() {
1024                                public void actionPerformed(ActionEvent lv_e) {
1025
1026        createPage(((JMenuItem)(lv_e.getSource())).getText());
```

```
                                    }
                              });
                        }

                  } catch (java.lang.Throwable ivjExc) {
                        handleException(ivjExc);
                  }
            }
            return iv_pageTypeMenu;
      }
      /**
       * Insert the method's description here.
       * Creation date: (9/29/99 11:36:37 AM)
       */
      public DefaultListModel getPageTypeModel() {
            if (iv_pageTypeModel == null) {
                  iv_pageTypeModel = new DefaultListModel();
            }
            return iv_pageTypeModel;
      }
      /**
       * Set session ID
       * Creation date: (9/29/99 11:36:37 AM)
       */
      public String getSessionID() {
            return iv_sessionID;
      }
      /**
       * Insert the method's description here.
       * Creation date: (9/29/99 11:36:37 AM)
       */
      public JMenuItem getStatusbarMenuItem() {
            return iv_statusbarMenuItem;
      }
      /**
       * Return the StatusBarPane property value.
       * @return javax.swing.JPanel
       */
      private javax.swing.JPanel getStatusBarPane() {
            if (iv_statusBarPane == null) {
                  try {
                        iv_statusMsg = new javax.swing.JLabel();
                        iv_statusMsg.setName("StatusMsg");
                        iv_statusMsg.setBorder(new
      javax.swing.border.EtchedBorder());
                        setStatusMessage("Welcome to the Franklin Editor");

                        iv_statusBarPane = new javax.swing.JPanel();
                        iv_statusBarPane.setName("StatusBarPane");
                        iv_statusBarPane.setLayout(new java.awt.BorderLayout());
                        iv_statusBarPane.add(iv_statusMsg, "Center");
                  } catch (java.lang.Throwable ivjExc) {
                        handleException(ivjExc);
                  }
            }
            return iv_statusBarPane;
      }
```

```
/**
 * Insert the method's description here.
 * Creation date: (9/29/99 11:36:37 AM)
 */
public JMenuItem getToolbarMenuItem() {
      return iv_toolbarMenuItem;
}
/**
 * Return the ToolBarPane property value.
 * @return javax.swing.JToolBar
 */
private javax.swing.JToolBar getToolBarPane() {
      if (iv_toolBarPane == null) {
            try {
                  // new
                  iv_newButton = new javax.swing.JButton();
                  iv_newButton.setName("NewButton");
                  iv_newButton.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/images/new.gif")));
                  iv_newButton.setText("");
                  iv_newButton.setToolTipText("Create new fragment or page");
                  iv_newButton.setMargin(new java.awt.Insets(0, 0, 0, 0));

      iv_newButton.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER)
;

      iv_newButton.setVerticalTextPosition(javax.swing.SwingConstants.BOTTOM);
                  iv_newButton.addActionListener(new ActionListener() {
                        public void actionPerformed(ActionEvent lv_e) {
                              newFragmentOrPage();
                        }
                  });

                  // delete
                  iv_deleteButton = new javax.swing.JButton();
                  iv_deleteButton.setName("RemoveButton");
                  iv_deleteButton.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/images/delete.gif")));
                  iv_deleteButton.setText("");
                  iv_deleteButton.setToolTipText("Remove current fragment or
page");
                  iv_deleteButton.setMargin(new java.awt.Insets(0, 0, 0, 0));

      iv_deleteButton.setHorizontalTextPosition(javax.swing.SwingConstants.CENT
ER);

      iv_deleteButton.setVerticalTextPosition(javax.swing.SwingConstants.BOTTOM
);

                  iv_deleteButton.addActionListener(new ActionListener() {
                        public void actionPerformed(ActionEvent lv_e) {
                              removeCurrentFragmentOrPage();
                        }
                  });

                  // cut
                  iv_cutButton = new javax.swing.JButton();
```

```
1141                    iv_cutButton.setName("CutButton");
1142                    iv_cutButton.setIcon(new
1143  javax.swing.ImageIcon(getClass().getResource("/images/cut.gif")));
1144                    iv_cutButton.setText("");
1145                    iv_cutButton.setToolTipText("Cut");
1146                    iv_cutButton.setMargin(new java.awt.Insets(0, 0, 0, 0));
1147
1148          iv_cutButton.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER)
1149  ;
1150
1151          iv_cutButton.setVerticalTextPosition(javax.swing.SwingConstants.BOTTOM);
1152                    iv_cutButton.addActionListener(new ActionListener() {
1153                          public void actionPerformed(ActionEvent lv_e) {
1154                                editCut();
1155                          }
1156                    });
1157
1158                    // copy
1159                    iv_copyButton = new javax.swing.JButton();
1160                    iv_copyButton.setName("CopyButton");
1161                    iv_copyButton.setIcon(new
1162  javax.swing.ImageIcon(getClass().getResource("/images/copy.gif")));
1163                    iv_copyButton.setText("");
1164                    iv_copyButton.setToolTipText("Copy");
1165                    iv_copyButton.setMargin(new java.awt.Insets(0, 0, 0, 0));
1166
1167          iv_copyButton.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER
1168  );
1169
1170          iv_copyButton.setVerticalTextPosition(javax.swing.SwingConstants.BOTTOM);
1171                    iv_copyButton.addActionListener(new ActionListener() {
1172                          public void actionPerformed(ActionEvent lv_e) {
1173                                editCopy();
1174                          }
1175                    });
1176
1177                    // paste
1178                    iv_pasteButton = new javax.swing.JButton();
1179                    iv_pasteButton.setName("PasteButton");
1180                    iv_pasteButton.setIcon(new
1181  javax.swing.ImageIcon(getClass().getResource("/images/paste.gif")));
1182                    iv_pasteButton.setText("");
1183                    iv_pasteButton.setToolTipText("Paste");
1184                    iv_pasteButton.setMargin(new java.awt.Insets(0, 0, 0, 0));
1185
1186          iv_pasteButton.setHorizontalTextPosition(javax.swing.SwingConstants.CENTE
1187  R);
1188
1189          iv_pasteButton.setVerticalTextPosition(javax.swing.SwingConstants.BOTTOM)
1190  ;
1191                    iv_pasteButton.addActionListener(new ActionListener() {
1192                          public void actionPerformed(ActionEvent lv_e) {
1193                                editPaste();
1194                          }
1195                    });
1196
1197                    // clear
```

```
1198                    iv_clearButton = new javax.swing.JButton();
1199                    iv_clearButton.setName("ClearButton");
1200                    iv_clearButton.setText("");
1201                    iv_clearButton.setToolTipText("Clear contents of edit form");
1202
1203          iv_clearButton.setHorizontalTextPosition(javax.swing.SwingConstants.CENTE
1204    R);
1205
1206          iv_clearButton.setVerticalTextPosition(javax.swing.SwingConstants.BOTTOM)
1207    ;
1208                    iv_clearButton.setIcon(new
1209    javax.swing.ImageIcon(getClass().getResource("/images/clear.gif")));
1210                    iv_clearButton.setMargin(new java.awt.Insets(0, 0, 0, 0));
1211
1212                    iv_clearButton.addActionListener(new ActionListener() {
1213                          public void actionPerformed(ActionEvent lv_e) {
1214                                clearCurrentFragmentOrPage();
1215                          }
1216                    });
1217
1218                    // back
1219                    iv_backButton = new javax.swing.JButton();
1220                    iv_backButton.setName("BackButton");
1221                    iv_backButton.setText("");
1222                    iv_backButton.setToolTipText("Cycle back through
1223    fragments/pages");
1224
1225          iv_backButton.setHorizontalTextPosition(javax.swing.SwingConstants.CENTER
1226    );
1227
1228          iv_backButton.setVerticalTextPosition(javax.swing.SwingConstants.BOTTOM);
1229                    iv_backButton.setIcon(new
1230    javax.swing.ImageIcon(getClass().getResource("/images/back.gif")));
1231                    iv_backButton.setMargin(new java.awt.Insets(0, 0, 0, 0));
1232
1233                    iv_backButton.addActionListener(new ActionListener() {
1234                          public void actionPerformed(ActionEvent lv_e) {
1235                                back();
1236                          }
1237                    });
1238
1239                    // forward
1240                    iv_forwardButton = new javax.swing.JButton();
1241                    iv_forwardButton.setName("ForwardButton");
1242                    iv_forwardButton.setText("");
1243                    iv_forwardButton.setToolTipText("Cycle forward through
1244    fragments/pages");
1245
1246          iv_forwardButton.setHorizontalTextPosition(javax.swing.SwingConstants.CEN
1247    TER);
1248
1249          iv_forwardButton.setVerticalTextPosition(javax.swing.SwingConstants.BOTTO
1250    M);
1251                    iv_forwardButton.setIcon(new
1252    javax.swing.ImageIcon(getClass().getResource("/images/forward.gif")));
1253                    iv_forwardButton.setMargin(new java.awt.Insets(0, 0, 0, 0));
1254
```

```
1255                    iv_forwardButton.addActionListener(new ActionListener() {
1256                          public void actionPerformed(ActionEvent lv_e) {
1257                                forward();
1258                          }
1259                    });
1260
1261                    // checkin
1262                    iv_checkinButton = new javax.swing.JButton();
1263                    iv_checkinButton.setName("CheckinButton");
1264                    iv_checkinButton.setText("");
1265                    iv_checkinButton.setToolTipText("Checkin the current
1266 fragment/page");
1267
1268       iv_checkinButton.setHorizontalTextPosition(javax.swing.SwingConstants.CEN
1269 TER);
1270
1271       iv_checkinButton.setVerticalTextPosition(javax.swing.SwingConstants.BOTTO
1272 M);
1273                    iv_checkinButton.setIcon(new
1274 javax.swing.ImageIcon(getClass().getResource("/images/checkin.gif")));
1275                    iv_checkinButton.setMargin(new java.awt.Insets(0, 0, 0, 0));
1276
1277                    iv_checkinButton.addActionListener(new ActionListener() {
1278                          public void actionPerformed(ActionEvent lv_e) {
1279                                checkin();
1280                          }
1281                    });
1282
1283                    // checkin
1284                    iv_checkoutButton = new javax.swing.JButton();
1285                    iv_checkoutButton.setName("CheckoutButton");
1286                    iv_checkoutButton.setText("");
1287                    iv_checkoutButton.setToolTipText("Checkout the selected
1288 fragment/page");
1289
1290       iv_checkoutButton.setHorizontalTextPosition(javax.swing.SwingConstants.CE
1291 NTER);
1292
1293       iv_checkoutButton.setVerticalTextPosition(javax.swing.SwingConstants.BOTT
1294 OM);
1295                    iv_checkoutButton.setIcon(new
1296 javax.swing.ImageIcon(getClass().getResource("/images/checkout.gif")));
1297                    iv_checkoutButton.setMargin(new java.awt.Insets(0, 0, 0, 0));
1298
1299                    iv_checkoutButton.addActionListener(new ActionListener() {
1300                          public void actionPerformed(ActionEvent lv_e) {
1301                                checkout();
1302                          }
1303                    });
1304
1305                    // create toolbar
1306                    iv_toolBarPane = new javax.swing.JToolBar();
1307                    iv_toolBarPane.setName("ToolBarPane");
1308                    iv_toolBarPane.add(iv_newButton,
1309 iv_newButton.getName());
1310                    iv_toolBarPane.add(iv_deleteButton,
1311 iv_deleteButton.getName());
```

```
1312                    iv_toolBarPane.addSeparator();
1313                    iv_toolBarPane.add(iv_cutButton,
1314    iv_cutButton.getName());
1315                    iv_toolBarPane.add(iv_copyButton,
1316    iv_copyButton.getName());
1317                    iv_toolBarPane.add(iv_pasteButton,
1318    iv_pasteButton.getName());
1319                    iv_toolBarPane.add(iv_clearButton,
1320    iv_clearButton.getName());
1321                    iv_toolBarPane.addSeparator();
1322                    iv_toolBarPane.add(iv_backButton,
1323    iv_backButton.getName());
1324                    iv_toolBarPane.add(iv_forwardButton,
1325    iv_forwardButton.getName());
1326                    iv_toolBarPane.addSeparator();
1327                    iv_toolBarPane.add(iv_checkinButton,
1328    iv_checkinButton.getName());
1329                    iv_toolBarPane.add(iv_checkoutButton,
1330    iv_checkoutButton.getName());
1331
1332              } catch (java.lang.Throwable ivjExc) {
1333                    handleException(ivjExc);
1334              }
1335        }
1336        return iv_toolBarPane;
1337    }
1338    /**
1339     * Return the ViewMenu property value.
1340     * @return javax.swing.JMenu
1341     */
1342    /* WARNING: THIS METHOD WILL BE REGENERATED. */
1343    private javax.swing.JMenu getViewMenu() {
1344          if (iv_viewMenu == null) {
1345                try {
1346                      iv_viewMenu = new javax.swing.JMenu();
1347                      iv_viewMenu.setName("ViewMenu");
1348                      iv_viewMenu.setText("View");
1349
1350                      // StatusbarMenuItem
1351                      iv_statusbarMenuItem = new javax.swing.JMenuItem();
1352                      iv_statusbarMenuItem.setName("StatusbarMenuItem");
1353                      iv_statusbarMenuItem.setText("Statusbar");
1354                      iv_statusbarMenuItem.addActionListener(new ActionListener() {
1355                            public void actionPerformed(ActionEvent lv_e) {
1356                                  viewStatusBar();
1357                            }
1358                      });
1359
1360                      // ToolbarMenuItem
1361                      iv_toolbarMenuItem = new javax.swing.JMenuItem();
1362                      iv_toolbarMenuItem.setName("ToolbarMenuItem");
1363                      iv_toolbarMenuItem.setText("Toolbar");
1364                      iv_toolbarMenuItem.addActionListener(new ActionListener() {
1365                            public void actionPerformed(ActionEvent lv_e) {
1366                                  viewToolBar();
1367                            }
1368                      });
```

```
1369
1370                    // ForwardMenuItem
1371                    iv_forwardMenuItem = new javax.swing.JMenuItem();
1372                    iv_forwardMenuItem.setName("ForwardMenuItem");
1373                    iv_forwardMenuItem.setText("Forward");
1374                    iv_forwardMenuItem.addActionListener(new ActionListener() {
1375                          public void actionPerformed(ActionEvent lv_e) {
1376                                forward();
1377                          }
1378                    });
1379
1380                    // BackMenuItem
1381                    iv_backMenuItem = new javax.swing.JMenuItem();
1382                    iv_backMenuItem.setName("BackMenuItem");
1383                    iv_backMenuItem.setText("Back");
1384                    iv_backMenuItem.addActionListener(new ActionListener() {
1385                          public void actionPerformed(ActionEvent lv_e) {
1386                                back();
1387                          }
1388                    });
1389
1390                    // HistoryMenu
1391                    iv_historyMenu = new javax.swing.JMenu();
1392                    iv_historyMenu.setName("historyMenu");
1393                    iv_historyMenu.setText("History");
1394
1395                    iv_viewMenu.add(iv_statusbarMenuItem);
1396                    iv_viewMenu.add(iv_toolbarMenuItem);
1397                    iv_viewMenu.add(new JSeparator());
1398                    iv_viewMenu.add(iv_backMenuItem);
1399                    iv_viewMenu.add(iv_forwardMenuItem);
1400                    iv_viewMenu.add(iv_historyMenu);
1401
1402              } catch (java.lang.Throwable ivjExc) {
1403                    handleException(ivjExc);
1404              }
1405        }
1406        return iv_viewMenu;
1407  }
1408  /**
1409   * Called whenever the part throws an exception.
1410   * @param exception java.lang.Throwable
1411   */
1412  private void handleException(java.lang.Throwable lv_exception) {
1413        /* Uncomment the following lines to print uncaught exceptions to stdout
1414  */
1415        System.out.println("--------- FranklinEditor.handleException---------");
1416        lv_exception.printStackTrace();
1417
1418  }
1419  /**
1420   * Initialize the Franklin Editor UI. called from constructor of FranklinEditor
1421   */
1422  public void initialize(TXDocument lv_initDocument) {
1423        try {
1424              Dispatcher.setFragmentAndPageTypes(this, lv_initDocument);  //
1425  setup dtd related items
```

```
1426              iv_fragmentManager = new FragmentManager(this);   // create
1427  fragment manager
1428              setName("FranklinEditor");
1429
1430        setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
1431  // need to logout +++++
1432              setJMenuBar(getFranklinEditorJMenuBar());         // menu bar
1433              setTitle("Franklin Editor");                      // window title
1434              setContentPane(getJFrameContentPane());           // now create the
1435  ui widgets after dtd set
1436
1437              Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
1438  // we call this twice (in main)
1439              /* Create the frame */
1440              pack(); // Pack frame on the screen
1441              setSize(800, 600);
1442
1443              /* Center frame on the screen */
1444              Dimension frameSize = getSize();
1445              if (frameSize.height > screenSize.height)
1446                    frameSize.height = screenSize.height;
1447              if (frameSize.width > screenSize.width)
1448                    frameSize.width = screenSize.width;
1449              setLocation((screenSize.width  - frameSize.width)  / 2,
1450                          (screenSize.height - frameSize.height) / 2);
1451              setVisible(true);
1452
1453              /* Add a windowListener for the windowClosingEvent */
1454              addWindowListener(new java.awt.event.WindowAdapter() {
1455                    public void windowClosing(java.awt.event.WindowEvent e) {
1456                          System.exit(0);
1457                    };
1458              });
1459        }
1460        catch (java.lang.Throwable ivjExc) {
1461              handleException(ivjExc);
1462        }
1463  }
1464  /**
1465   * Insert the method's description here.
1466   * Creation date: (10/11/99 6:25:59 PM)
1467   */
1468  public void loadAllDTDs() {
1469        Dispatcher.loadAllDTDs(this);
1470  }
1471  /**
1472   * Insert the method's description here.
1473   * Creation date: (10/7/99 9:27:02 AM)
1474   */
1475  public static void loadProperties() {
1476        Properties lv_props = new Properties();
1477        try {
1478              InputStream is =
1479  ClassLoader.getSystemResourceAsStream("franklin.properties");
1480              lv_props.load(is);
1481              is.close();
```

```
1482              System.out.println("Loaded properties from franklin.properties: " +
1483  lv_props);
1484
1485              Class.forName("com.ibm.adtech.franklin.client.Dispatcher");     //
1486  make sure Dispatcher class is loaded
1487              String lv_value;
1488
1489              // cv_standaloneP   = "true"
1490              lv_value = (String)(lv_props.get("standaloneP"));
1491              if ((lv_value != null) && !(lv_value.equals(""))) {
1492                    System.out.println("Property cv_standaloneP = " + lv_value);
1493                    cv_standaloneP = (Boolean.valueOf(lv_value)).booleanValue();
1494              }
1495
1496              // cv_splashscreenP = "false"
1497              //lv_value = (String)(lv_props.get("splashscreenP"));
1498              //if ((lv_value != null) && !(lv_value.equals(""))) {
1499              //    System.out.println("Property cv_splashscreenP = " +
1500  lv_value);
1501              //    cv_splashscreenP =
1502  (Boolean.valueOf(lv_value)).booleanValue();
1503              //}
1504
1505              // cv_server        =
1506  "http://frasier.dhcp.adtech.internet.ibm.com/franklin/";
1507              //lv_value = (String)(lv_props.get("server"));
1508              //if ((lv_value != null) && !(lv_value.equals(""))) {
1509              //    System.out.println("Property cv_server = " + lv_value);
1510              //    Dispatcher.cv_server = lv_value;
1511              //}
1512
1513              // cv_dispatcher    = "http://9.242.61.42/franklin";        //
1514  dikran's server in southbury
1515              lv_value = (String)(lv_props.get("dispatcher"));
1516              if ((lv_value != null) && !(lv_value.equals(""))) {
1517                    System.out.println("Property cv_dispatcher = " + lv_value);
1518                    Dispatcher.cv_dispatcher = lv_value;
1519              }
1520
1521              // cv_initXMLFile   = "/xml/initXMLFile.xml";
1522              lv_value = (String)(lv_props.get("initXMLFile"));
1523              if ((lv_value != null) && !(lv_value.equals(""))) {
1524                    System.out.println("Property cv_initXMLFile = " + lv_value);
1525                    Dispatcher.cv_initXMLFile = lv_value;
1526              }
1527        }
1528        catch (ClassNotFoundException lv_e) {
1529              System.out.println("loadProperties, ClassNotFoundException");
1530              lv_e.printStackTrace();
1531        }
1532        catch (IOException lv_e) {
1533              System.out.println("loadProperties, IOException");
1534              lv_e.printStackTrace();
1535        }
1536        catch (NullPointerException lv_e) {
1537               System.out.println("loadProperties, NullPointerException");
1538               lv_e.printStackTrace();
```

```
1539        }
1540
1541  }
1542  /**
1543   * Starts the application.
1544   * @param args an array of command-line arguments
1545   */
1546  public static void main(java.lang.String[] args) {
1547        try {
1548
1549        //UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
1550  // Set native look and feel
1551
1552              loadProperties();                                          // load
1553  properties from "franklin.properties" file
1554              Dispatcher.initialize();                                   //
1555  initialize Dispatcher
1556              FranklinEditor lv_editor = new FranklinEditor();           // create
1557  shell of editor first.. to save sessionID
1558              lv_editor.iv_screenSize =
1559  Toolkit.getDefaultToolkit().getScreenSize(); // Calculate the screen size
1560              new FranklinEditorLoginScreen(lv_editor);
1561  // create Login Screen
1562        }
1563        catch (Throwable exception) {
1564              System.err.println("Exception occurred in main() of
1565  FranklinEditor");
1566              exception.printStackTrace(System.out);
1567        }
1568  }
1569  /**
1570   * put up dialog to create a new fragment or page
1571   * Creation date: (9/29/99 11:36:37 AM)
1572   */
1573  public void newFragmentOrPage() {
1574        System.out.println("newFragmentOrPage");
1575        if (iv_typeDialog == null) {
1576              iv_typeDialog = new FranklinEditorTypeDialog(this, true);
1577        }
1578        iv_typeDialog.setVisible(true);
1579  }
1580  /**
1581   * Insert the method's description here.
1582   * Creation date: (10/7/99 2:22:35 PM)
1583   * @param lv_string java.lang.String
1584   */
1585  public static void printDebug(String lv_string) {
1586        if (cv_debug)
1587              System.out.println(lv_string);
1588  }
1589  /**
1590   * This is called after a fragment or page is created or the user
1591   * clicks on the history back/forward buttons.
1592   * The method replaces the pane in the right side of the editor.
1593   * Creation date: (10/7/99 7:06:54 PM)
1594   */
1595  public void redisplayEditorPane() {
```

```
1596        ClientFragment lv_fragment = iv_fragmentManager.getCurrentFragment();
1597        if (lv_fragment != null) {
1598              if (lv_fragment.iv_pane != null) {
1599                    Container lv_parent = iv_editorPane.getParent();
1600                    lv_parent.remove(iv_editorPane);
1601                    lv_parent.add(lv_fragment.iv_pane, "right");
1602                    iv_editorPane = lv_fragment.iv_pane;
1603                    iv_editorPane.invalidate();
1604                    doLayout();
1605                    setStatusMessage("Switching to fragment " + lv_fragment);
1606              }
1607              else {
1608                    setStatusMessage("No interface exists for fragment " +
1609  lv_fragment);
1610              }
1611        }
1612        else if (iv_editorPane != iv_noPane) {
1613              // remove the last pane and replace with one that indicates no
1614  current element
1615              Container lv_parent = iv_editorPane.getParent();
1616              lv_parent.remove(iv_editorPane);
1617              lv_parent.add(iv_noPane, "right");
1618              iv_editorPane = iv_noPane;
1619              iv_editorPane.invalidate();
1620              doLayout();
1621              setStatusMessage("History list is empty");
1622        }
1623  }
1624  /**
1625   * Insert the method's description here.
1626   * Creation date: (10/11/99 6:07:36 PM)
1627   */
1628  private void reloadDTDs() {
1629        iv_fragmentTypeModel.clear();
1630        iv_pageTypeModel.clear();
1631        Dispatcher.reloadDTDs(this);
1632  }
1633  /**
1634   * remove the current fragment or page from the UI
1635   * may need to unlock fragment on server
1636   * // +++ dialog to confirm deletion
1637   * Creation date: (9/29/99 11:36:37 AM)
1638   */
1639  public void removeCurrentFragmentOrPage() {
1640        System.out.println("removeCurrentFragmentOrPage");
1641        ClientFragment lv_fragment = iv_fragmentManager.getCurrentFragment();
1642        if (lv_fragment != null) {
1643              if ((lv_fragment.getLock() != null) &&
1644                    (!(lv_fragment.getLock().equals("")))) {
1645                    boolean lv_status = Dispatcher.unlock(getSessionID(),
1646  lv_fragment);
1647                    // this setStatusMessage will be overwritten by redisplay
1648  message
1649                    if (lv_status) setStatusMessage("Fragment " + lv_fragment + "
1650  unlocked");
1651                    else setStatusMessage("Error unlocking fragment " +
1652  lv_fragment);
```

```
1653              }
1654              iv_fragmentManager.removeFragment(lv_fragment);
1655              iv_historyMenu.remove(lv_fragment.getMenuItem());     // remove from
1656  history menu
1657              redisplayEditorPane();                                // redisplay
1658  current ClientFragment
1659        }
1660  }
1661  /**
1662   * remove a fragment or page from the UI
1663   * may need to unlock fragment on server
1664   * Creation date: (9/29/99 11:36:37 AM)
1665   */
1666  public void removeFragmentOrPage(ClientFragment lv_fragment) {
1667        System.out.println("removeFragmentOrPage");
1668        if (lv_fragment != null) {
1669              // +++ dialog to confirm deletion
1670              //String lv_status = Dispatcher.unlockFragment(lv_fragment);
1671              //setStatusMessage(lv_status);                          // this will
1672  be overwritten by redisplay message
1673              iv_fragmentManager.removeFragment(lv_fragment);
1674              iv_historyMenu.remove(lv_fragment.getMenuItem());     // remove from
1675  history menu
1676              redisplayEditorPane();                                // redisplay
1677  current ClientFragment
1678        }
1679  }
1680  /**
1681   * Insert the method's description here.
1682   * Creation date: (10/8/99 11:15:52 AM)
1683   */
1684  public void search() {
1685        setStatusMessage("No search yet");
1686  }
1687  /**
1688   * Set session ID
1689   * Creation date: (9/29/99 11:36:37 AM)
1690   */
1691  public void setSessionID(String lv_id) {
1692        iv_sessionID = lv_id;
1693  }
1694  /**
1695   * Insert the method's description here.
1696   * Creation date: (9/29/99 11:36:37 AM)
1697   */
1698  public void setStatusMessage(String lv_message) {
1699        iv_statusMsg.setText(lv_message);
1700  }
1701  public void showAboutBox() {
1702        /* Create the AboutBox dialog */
1703        FranklinEditorAboutBox aFranklinEditorAboutBox = new
1704  FranklinEditorAboutBox();
1705        Dimension dialogSize = aFranklinEditorAboutBox.getPreferredSize();
1706        Dimension frameSize = getSize();
1707        Point loc = getLocation();
1708        aFranklinEditorAboutBox.setLocation((frameSize.width - dialogSize.width)
1709  / 2 + loc.x, (frameSize.height - dialogSize.height) / 2 + loc.y);
```

```
1710        aFranklinEditorAboutBox.setModal(true);
1711        aFranklinEditorAboutBox.show();
1712    }
1713    /**
1714     * Insert the method's description here.
1715     * Creation date: (10/8/99 11:15:52 AM)
1716     */
1717    public void showHelpTopics() {
1718        setStatusMessage("No Help topics yet");
1719    }
1720    /**
1721     * Given a message string, show this in a modal dialog
1722     * @param lv_message java.lang.String
1723     */
1724    public void showMessageDialog(String lv_message, int lv_messageType) {
1725        JOptionPane.showMessageDialog(this, lv_message, "Franklin Editor Message",
1726    lv_messageType);
1727    }
1728    /**
1729     * This method was created in VisualAge.
1730     * @param lv_count int
1731     */
1732    public void updateCount(int lv_count) {
1733        String lv_stringCount = Integer.toString(lv_count);
1734        iv_countLabel.setText(lv_stringCount + " ");
1735    }
1736    public void viewStatusBar() {
1737        /* Hide or show the statusbar */
1738        getStatusBarPane().setVisible(!(getStatusBarPane().isVisible()));
1739    }
1740    public void viewToolBar() {
1741        /* Hide or show the toolbar */
1742        getToolBarPane().setVisible(!(getToolBarPane().isVisible()));
1743    }
1744    }
```

```
1  package com.ibm.adtech.franklin.client;
2
3  import org.w3c.dom.NodeList;
4  import org.w3c.dom.Node;
5  import com.ibm.xml.parser.AttDef;
6  import com.ibm.xml.parser.ContentModel;
7  import com.ibm.xml.parser.CMNode;
8  import com.ibm.xml.parser.CM1op;
9  import com.ibm.xml.parser.CM2op;
10 import com.ibm.xml.parser.CMLeaf;
11 import com.ibm.xml.parser.DTD;
12 import com.ibm.xml.parser.InsertableElement;
13 import com.ibm.xml.parser.ElementDecl;
14 import com.ibm.xml.parser.TXDocument;
15 import com.ibm.xml.parser.TXElement;
16 import java.util.Enumeration;
17 import java.util.Vector;
18 import java.util.Hashtable;
19 import javax.swing.*;
20 import javax.swing.JPanel;
21 import javax.swing.text.*;
22 import java.awt.*;
23 import java.awt.BorderLayout;
24 //import java.awt.event.TextListener;
25 import javax.swing.event.DocumentListener;
26 import javax.swing.event.DocumentEvent;
27 //import javax.swing.event.TextListener;
28 import java.awt.event.ActionEvent;
29
30 /**
31  * Insert the type's description here.
32  * Creation date: (10/11/99 1:29:12 PM)
33  * @author:
34  */
35 public class InterfaceMaker {
36
37 /**
38  * for SKIP elements, we want to process children only to put them into the
39 Document
40  * no ui to worry about here. we know we are under a SKIP node
41  * we have to make sure we don't add PCDATA nodes to tree.
42  *
43  * Creation date: (11/04/99)
44  */
45 public static void createInterfaceElementsOnly(DTD lv_dtd, CMNode lv_cmNode,
46 TXDocument lv_document, Node lv_docElement) {
47       //System.out.println("createInterfaceElementsOnly (" + lv_cmNode + ")");
48       String lv_labelBase = lv_cmNode.toString();
49
50       if (lv_cmNode == null) {                                          // not
51 a MODEL_GROUP, ie ANY or EMPTY
52             System.out.println("createInterfaceElementsOnly: not a model_group
53 " + lv_cmNode);
54       }
55       else { // this could be a #PCDATA node or some Element. we don't want to
56 create UIs for PCDATA
```

```
57              if ((lv_cmNode instanceof CM1op) && ((CM1op)lv_cmNode).getNode()
58  instanceof CMLeaf) {  // with qualifiers
59                      CMNode lv_tmpNode = ((CM1op)lv_cmNode).getNode();
60                      if (!((CMLeaf)lv_tmpNode).getName().equals("#PCDATA")) {
61                              createInterfaceElementsOnlyAux(lv_dtd, lv_cmNode,
62  lv_document, lv_docElement);
63                      }
64              }
65              else if (lv_cmNode instanceof CMLeaf)
66  {                                                   // no qualifiers
67                      if (!((CMLeaf)lv_cmNode).getName().equals("#PCDATA")) {
68                              createInterfaceElementsOnlyAux(lv_dtd, lv_cmNode,
69  lv_document, lv_docElement);
70                      }
71              }
72              else
73  {
74     // go recursive
75                      createInterfaceElementsOnly(lv_dtd,
76  ((CM2op)lv_cmNode).getLeft(), lv_document, lv_docElement);
77                      createInterfaceElementsOnly(lv_dtd,
78  ((CM2op)lv_cmNode).getRight(), lv_document, lv_docElement);
79              }
80      }
81  }
82  /**
83   * for SKIP elements, we want to process children only to put them into the
84  Document
85   * no ui to worry about here. we know we are under a SKIP node
86   * we have to make sure we don't add PCDATA nodes to tree.
87   *
88   * Creation date: (11/04/99)
89   */
90  public static void createInterfaceElementsOnlyAux(DTD lv_dtd, CMNode lv_cmNode,
91  TXDocument lv_document, Node lv_docElement) {
92        //System.out.println("createInterfaceElementsOnlyAux (" + lv_cmNode +
93  ")");
94        String lv_labelBase = "";
95        if ((lv_cmNode instanceof CM1op) && ((CM1op)lv_cmNode).getNode()
96  instanceof CMLeaf) {
97              lv_labelBase = ((CM1op)lv_cmNode).getNode().toString();
98        }
99        else {
100             lv_labelBase = lv_cmNode.toString();
101       }
102
103       TXElement lv_newElement =
104 (TXElement)(lv_document.createElement(lv_labelBase));
105       lv_docElement.appendChild(lv_newElement);
106       lv_docElement = lv_newElement;
107       ContentModel lv_contentModel = lv_dtd.getContentModel(lv_labelBase);
108       if (lv_contentModel != null) {
109             CMNode lv_contentModelNode = lv_contentModel.getContentModelNode();
110             createInterfaceElementsOnly(lv_dtd, lv_contentModelNode,
111 lv_document, lv_docElement);
112       }
113 }
```

```
/**
 * For the given DTD and content model node, create appropriate input widgets
and add to the JPanel.
 * Make sure that the docElement stays in sync with the widgets in the UI.
 * The docElement is the parent of the children described in the model sent in.
 * <br>
 * This means look at the attribute DATATYPE on the node and see what the type
is (one of uitypes.txt)
 * DATE | INTEGER | STRING | SHORTTEXT | LONGTEXT | CHOICE
 * Also, we need to check whether the element has children and if the children
should be displayed
 * For example, BODY has PARAGRAPH children. but the paragraph children are not
displayed.
 * For example, RELATEDLINK has children URL and LINKTITLE.
 * Here the relatedlink is only a title. URL and LINKTITLE are subtitles and
have UI input widgets.
 * Also, associate that element in the xml DOM with the newly created input
widget.
 *
 * Creation date: (10/11/99 1:31:41 PM)
 */
public static void createInterfaceForModel(DTD lv_dtd, ClientFragment
lv_fragment, JPanel lv_pane, CMNode lv_cmNode, TXDocument lv_document, Node
lv_docElement) {

      if (lv_cmNode == null) {                                          // not
a MODEL_GROUP, ie ANY or EMPTY
            System.out.println("createInterfaceForModel: not a model_group " +
lv_cmNode);
      }
      else {
            if ((lv_cmNode instanceof CM1op) && ((CM1op)lv_cmNode).getNode()
instanceof CMLeaf) {  // has qualifiers '*' or '+' or '?'
                  createInterfaceWithQualifiers(lv_dtd, lv_fragment, lv_pane,
lv_cmNode, lv_document, lv_docElement);
            }
            else if (lv_cmNode instanceof CMLeaf) {
                  // this could be a #PCDATA node or some Element. we don't
want to create UIs for PCDATA
                  if (!(((CMLeaf)lv_cmNode).getName().equals("#PCDATA"))) {
                        createInterfaceWithoutQualifiers(lv_dtd, lv_fragment,
lv_pane, lv_cmNode, lv_document, lv_docElement); // no qualifiers (*, + or ?)
                  }
            }
            else
{
   // go recursive
                  createInterfaceForModel(lv_dtd, lv_fragment, lv_pane,
((CM2op)lv_cmNode).getLeft(), lv_document, lv_docElement);
                  createInterfaceForModel(lv_dtd, lv_fragment, lv_pane,
((CM2op)lv_cmNode).getRight(), lv_document, lv_docElement);
            }
      }
}
/**
 * We have a content model without any qualifiers. (no *, + or ?)
 * Add child node to docElement.
```

```
 * Link docElement with UI
 *
 * Creation date: (11/04/99)
 */
public static void createInterfaceWithoutQualifiers(DTD lv_dtd, ClientFragment
lv_fragment, JPanel lv_pane, CMNode lv_cmNode, TXDocument lv_document, Node
lv_docElement) {
      System.out.println("ContentType 2 is (" + lv_cmNode + ")");
      String lv_label     = "";
      String lv_labelBase = lv_cmNode.toString();
      JPanel lv_myPane    = new JPanel();
      JLabel lv_labelPane = new JLabel("", SwingConstants.RIGHT);
      JComponent lv_component = null;
      lv_myPane.setLayout(new BorderLayout());
      lv_labelPane.setVerticalAlignment(SwingConstants.TOP);
      lv_labelPane.setFont(FranklinEditor.cv_labelFont);
      lv_labelPane.setPreferredSize(new
Dimension(FranklinEditor.cv_fragmentLabelWidth,
FranklinEditor.cv_fragmentTitleHeight));

      // create the element and add to document
      TXElement lv_newElement =
(TXElement)(lv_document.createElement(lv_labelBase));
      lv_docElement.appendChild(lv_newElement);

      // skip if SYSTEM or SPECIAL (only need to check SPECIAL here cus its
without a qualifier)
      if (!skipElement(lv_labelBase) && !skipNonDisplayElement(lv_labelBase)) {
            lv_label = lv_labelBase + " (req)";
            lv_labelPane.setText(lv_label);
            lv_pane.add(lv_myPane);
            lv_myPane.add(lv_labelPane, "West");
            lv_component = getInputWidget(lv_dtd, lv_fragment, lv_newElement,
true);
            lv_component.setBackground(Color.red);
      }
      if (lv_component == null) {            // if we skipped it or no widget
at this level, go into model of children
            ContentModel lv_contentModel =
lv_dtd.getContentModel(lv_labelBase);
            if (lv_contentModel != null) {
                  CMNode lv_contentModelNode =
lv_contentModel.getContentModelNode();
                  if (skipElement(lv_labelBase))
{                                    // for SYSTEM
                        createInterfaceElementsOnly(lv_dtd,  lv_contentModelNode,
lv_document, lv_newElement);
                  }
                  else {
                        createInterfaceForModel(lv_dtd, lv_fragment, lv_pane,
lv_contentModelNode, lv_document, lv_newElement);
                  }
            }
      }
      else {
            lv_myPane.add(lv_component, "Center");
      }
```

```
}
/**
 * We have a content model that contains qualifiers '*' or '+' or '?'
 *
 * Then associate that element in the xml DOM with the newly created input
widget
 * Creation date: (10/11/99 1:31:41 PM)
 */
public static void createInterfaceWithQualifiers(DTD lv_dtd, ClientFragment
lv_fragment, JPanel lv_pane, CMNode lv_cmNode, TXDocument lv_document, Node
lv_docElement) {
      String lv_label = "";
      String lv_labelBase = "";
      String lv_qualifier = "";
      JPanel lv_myPane = new JPanel();
      JLabel lv_labelPane = new JLabel("", SwingConstants.RIGHT);
      JComponent lv_component = null;
      lv_myPane.setLayout(new BorderLayout());
      lv_labelPane.setVerticalAlignment(SwingConstants.TOP);
      lv_labelPane.setFont(FranklinEditor.cv_labelFont);
      lv_labelPane.setPreferredSize(new
Dimension(FranklinEditor.cv_fragmentLabelWidth,
FranklinEditor.cv_fragmentTitleHeight));

      CM1op lv_cs1 = (CM1op)lv_cmNode;
      System.out.println("ContentType 1 is (" + lv_cs1.getNode() + ")" + (char)
lv_cs1.getType());
      lv_labelBase = lv_cs1.getNode().toString();
      lv_qualifier = String.valueOf((char) lv_cs1.getType());

      // create the element and add to document
      TXElement lv_newElement =
(TXElement)(lv_document.createElement(lv_labelBase));
      lv_docElement.appendChild(lv_newElement);

      if (!skipElement(lv_labelBase)) {                          // is this an
element to skip
            lv_component = getInputWidget(lv_dtd, lv_fragment, lv_newElement,
false); // returns null if no DATATYPE attribute
            if (lv_qualifier.equals("*")) {
                  lv_label = lv_labelBase + " (0+)";
                  MoreOrLess lv_moreOrLess = new MoreOrLess(lv_labelBase,
lv_newElement, lv_component);
                  lv_myPane.add(lv_moreOrLess, "Center");
            }
            else if (lv_qualifier.equals("+")) {                  //
                        lv_label = lv_labelBase + " (1+)";
                        MoreOrLess lv_moreOrLess = new MoreOrLess(lv_labelBase,
lv_newElement, lv_component);
                        lv_myPane.add(lv_moreOrLess, "Center");
            }
            else if (lv_qualifier.equals("?")) {
                  lv_label = lv_labelBase + " (opt)";
                  if (!(lv_component == null)) lv_myPane.add(lv_component,
"Center");
            }
            else {
```

```
285                     lv_label = lv_labelBase + (char) lv_cs1.getType();  // ???
286 ++++
287                     if (!(lv_component == null)) lv_myPane.add(lv_component,
288 "Center");
289             }
290             lv_pane.add(lv_myPane);                                  // add new
291 pane to ongoing set of widgets
292             lv_labelPane.setText(lv_label);
293             lv_myPane.add(lv_labelPane, "West");
294             if (lv_component == null) {                              // no
295 component for this level but children have component
296                     ContentModel lv_contentModel =
297 lv_dtd.getContentModel(lv_labelBase);
298                     if (lv_contentModel != null) {
299                             CMNode lv_contentModelNode =
300 lv_contentModel.getContentModelNode();
301                             if (skipElement(lv_labelBase))
302 {                                           // for CLIENT or SYSTEM
303                                     createInterfaceElementsOnly(lv_dtd,
304 lv_contentModelNode, lv_document, lv_newElement);
305                             }
306                             else {
307                                     createInterfaceForModel(lv_dtd, lv_fragment,
308 lv_pane, lv_contentModelNode, lv_document, lv_newElement);
309                             }
310                     }
311             }
312         }
313 }
314 /**
315  * return a JComponent that will be for the DTD element lv_elementName
316  * Only if there is an attribute of cv_DATATYPE do we actually want to create a
317 widget
318  * if there is an attribute called cv_CHOICES, then we need to produce a choice
319 widget
320  * also, save the widget in the ui hashtable. the key is the element from the
321 DOM
322  * Creation date: (10/12/99 12:50:12 PM)
323  * @param lv_dtd com.ibm.xml.parser.DTD
324  * @param lv_elementName java.lang.String
325  */
326 public static JComponent getInputWidget(DTD lv_dtd, ClientFragment lv_fragment,
327 Node lv_element, boolean lv_required) {
328         String      lv_elementName  = lv_element.getNodeName();
329         AttDef      lv_attDef       = null;
330         AttDef      lv_attChoiceDef = null;
331         JComponent  lv_component    = null;
332         JScrollPane lv_scroll       = null;
333
334         System.out.println("getInputWidget: lv_elementName=" + lv_elementName);
335         lv_attDef       = lv_dtd.getAttributeDeclaration(lv_elementName,
336 FranklinEditor.cv_DATATYPE);
337         lv_attChoiceDef = lv_dtd.getAttributeDeclaration(lv_elementName,
338 FranklinEditor.cv_CHOICES);
339
340         if (lv_attDef != null) {   // have ELEMENT that has DATATYPE attribute,
341 ie we want a UI widget
```

```
342             String lv_attValue = lv_attDef.getDefaultStringValue();
343             if (lv_attValue != null) {
344                   if (lv_attChoiceDef != null) {                        //
345 CHOICE
346                         Enumeration lv_elem = lv_attChoiceDef.elements();
347                         lv_component = getInputWidgetChoice(lv_elem,
348 lv_required);
349                   }
350                   else {                                                //
351 DATE, INTEGER, STRING, SHORTTEXT, LONGTEXT
352                         lv_component = getInputWidgetText(lv_attValue,
353 lv_required);
354                   }
355             }
356             else {                            // this shouldn't be run...only if
357 haven't accounted for attribute
358                   System.out.println("getInputWidget: *** Shouldn't be called,
359 value of DATATYPE for UI is null *** " + lv_elementName);
360             }
361             lv_fragment.putInterface(lv_element, lv_component);   // if
362 lv_component is not null add to UI hashtable
363             lv_scroll = new JScrollPane(lv_component);
364       }
365       return lv_scroll;
366 }
367 /**
368  * this getInputWidgetChoice is explicitly for the CHOICE datatype.
369  * return a JComboBox for this choice with all the possible choices added
370  * lv_required means that this widget is required in the DTD... so visually
371 distinquish it by
372  * changing its background
373  *
374  * Creation date: (10/12/99 12:50:12 PM)
375  * @param lv_dtd com.ibm.xml.parser.DTD
376  * @param lv_elementName java.lang.String
377  */
378 public static JComponent getInputWidgetChoice(Enumeration lv_choices, boolean
379 lv_required) {
380       //System.out.println("getInputWidgetChoice");
381       JComponent  lv_component = null;
382
383       DefaultComboBoxModel lv_model = new DefaultComboBoxModel();
384       while (lv_choices.hasMoreElements()) {
385             String lv_choice = (String)lv_choices.nextElement();
386             lv_model.addElement(lv_choice);
387       }
388       lv_component = new JComboBox(lv_model);
389       if (lv_required) {
390             lv_component.setBackground(FranklinEditor.cv_requiredInputColor);
391             ((JComboBox)lv_component).addActionListener(new
392 java.awt.event.ActionListener() {
393                   public void actionPerformed(ActionEvent lv_e) {
394                         // if the value is the first element of the combobox
395 (ie, its the default), then color it inputRequired
396                         DefaultComboBoxModel lv_model =
397 (DefaultComboBoxModel)((JComboBox)lv_e.getSource()).getModel();
```

```
398                         if(lv_model.getSelectedItem() ==
399 lv_model.getElementAt(0)) {
400
401       ((JComboBox)lv_e.getSource()).setBackground(FranklinEditor.cv_requiredInp
402 utColor);
403                         }
404                         else {
405
406       ((JComboBox)lv_e.getSource()).setBackground(FranklinEditor.cv_inputColor)
407 ;
408                         }
409                   }
410             });
411       }
412       return lv_component;
413 }
414 /**
415  * return a JComponent that corresponds to the DATATYPE default value lv_type
416  * this method only knows about DATE, INTEGER, STRING, SHORTTEXT, LONGTEXT
417  * added addDocumentListener to change highlighting of the widget if it is
418 required and user adds/deletes content
419  *
420  * Creation date: (10/12/99 12:50:12 PM)
421  * @param lv_dtd com.ibm.xml.parser.DTD
422  * @param lv_elementName java.lang.String
423  */
424 public static JComponent getInputWidgetText(String lv_uitype, boolean
425 lv_required) {
426       //System.out.println("getInputWidget: lv_type = " + lv_uitype);
427       JComponent  lv_component = null;
428
429       // we have a valid uitype one of DATE, INTEGER, STRING, SHORTTEXT,
430 LONGTEXT
431       if (lv_uitype != null && !(lv_uitype.equals("")))
432 {                                   // one liner
433             if (lv_uitype.equals("DATE")    ||
434                   lv_uitype.equals("INTEGER") ||
435                   lv_uitype.equals("STRING")) {
436                         lv_component = new JTextField();
437                         lv_component.setPreferredSize(new
438 Dimension(FranklinEditor.cv_fragmentTextWidth,
439
440 FranklinEditor.cv_fragmentTitleHeight));
441
442       lv_component.setBorder(BorderFactory.createLineBorder(Color.black));
443
444       ((JTextField)lv_component).getDocument().putProperty("View",
445 lv_component);
446                         if (lv_required) {
447
448       lv_component.setBackground(FranklinEditor.cv_requiredInputColor);
449
450       ((JTextField)lv_component).getDocument().addDocumentListener(new
451 DocumentListener() {
452                                     public void changedUpdate(DocumentEvent
453 lv_e) { }
```

```
454                                     public void insertUpdate(DocumentEvent
455   lv_e) {
456                                           if (lv_e.getDocument().getLength() >
457   0) {
458                                                 //System.out.println("clear
459   color to white " + FranklinEditor.cv_inputColor);
460
461         ((JTextField)lv_e.getDocument().getProperty("View")).setBackground(Frankl
462   inEditor.cv_inputColor);
463                                           }
464                                     }
465                                     public void removeUpdate(DocumentEvent
466   lv_e) {
467                                           if (lv_e.getDocument().getLength() ==
468   0) {
469                                                 //System.out.println("setting
470   color to red " + FranklinEditor.cv_requiredInputColor);
471
472         ((JTextField)lv_e.getDocument().getProperty("View")).setBackground(Frankl
473   inEditor.cv_requiredInputColor);
474                                           }
475                                     }
476                               });
477                         }
478               }
479               else if (lv_uitype.equals("SHORTTEXT")) {
480                     lv_component = new JTextArea(3,30);
481                     ((JTextArea)lv_component).setLineWrap(true);
482
483         lv_component.setBorder(BorderFactory.createLineBorder(Color.black));
484                     ((JTextArea)lv_component).getDocument().putProperty("View",
485   lv_component);
486                     if (lv_required) {
487
488         lv_component.setBackground(FranklinEditor.cv_requiredInputColor);
489
490         ((JTextArea)lv_component).getDocument().addDocumentListener(new
491   DocumentListener() {
492                                     public void changedUpdate(DocumentEvent
493   lv_e) { }
494                                     public void insertUpdate(DocumentEvent
495   lv_e) {
496                                           if (lv_e.getDocument().getLength() >
497   0) {
498
499         ((JTextArea)lv_e.getDocument().getProperty("View")).setBackground(Frankli
500   nEditor.cv_inputColor);
501                                           }
502                                     }
503                                     public void removeUpdate(DocumentEvent
504   lv_e) {
505                                           if (lv_e.getDocument().getLength() ==
506   0) {
507
508         ((JTextArea)lv_e.getDocument().getProperty("View")).setBackground(Frankli
509   nEditor.cv_requiredInputColor);
510                                           }
```

```
                                        }
                                });
                        }
                }
                else if (lv_uitype.equals("LONGTEXT"))
{                                       // large text area
                        lv_component = new JTextArea(30,30);
                        ((JTextArea)lv_component).setLineWrap(true);

        lv_component.setBorder(BorderFactory.createLineBorder(Color.black));
                        ((JTextArea)lv_component).getDocument().putProperty("View",
lv_component);
                        if (lv_required) {

        lv_component.setBackground(FranklinEditor.cv_requiredInputColor);

        ((JTextArea)lv_component).getDocument().addDocumentListener(new
DocumentListener() {
                                        public void changedUpdate(DocumentEvent
lv_e) { }
                                        public void insertUpdate(DocumentEvent
lv_e) {
                                                if (lv_e.getDocument().getLength() >
0) {

        ((JTextArea)lv_e.getDocument().getProperty("View")).setBackground(Frankli
nEditor.cv_inputColor);
                                                }
                                        }
                                        public void removeUpdate(DocumentEvent
lv_e) {
                                                if (lv_e.getDocument().getLength() ==
0) {

        ((JTextArea)lv_e.getDocument().getProperty("View")).setBackground(Frankli
nEditor.cv_requiredInputColor);
                                                }
                                        }
                                });
                        }
                }
                else {
                        System.out.println("getInputWidget: *** Unknown type *** " +
lv_uitype);
                        return null;
                }
                return lv_component;
        }
        return null;
}
/**
 * Given a dtd, return a JPanel that will allow the user to edit the fields.
 * the title coming in will be the name of the xml document to create
 * assumes that the dtd has a SYSTEM and CLIENT tag that are ignored.
 * and that it has a SPECIAL tag that are the elements special to this dtd type
 *
 * Creation date: (10/11/99 1:31:41 PM)
```

```
 */
public static JPanel getInterface(DTD lv_dtd, ClientFragment lv_fragment) {
      String      lv_stuff             = "";
      JPanel      lv_currentPanel      = new JPanel();
      JPanel      lv_currentInputPanel = new JPanel();
      JPanel      lv_titlePane         = new JPanel();
      JLabel      lv_title             = new JLabel(lv_fragment.toString(),
SwingConstants.RIGHT);
      lv_currentPanel.setLayout(new BorderLayout());
      lv_currentInputPanel.setLayout(new BoxLayout(lv_currentInputPanel,
BoxLayout.Y_AXIS));
      lv_title.setFont(FranklinEditor.cv_titleFont);
      lv_title.setPreferredSize(new
Dimension(FranklinEditor.cv_fragmentLabelWidth,
FranklinEditor.cv_fragmentTitleHeight));
      lv_titlePane.setLayout(new BorderLayout());
      lv_titlePane.add(lv_title, "West");
      lv_currentPanel.add(lv_titlePane, "North");
      lv_currentPanel.add(lv_currentInputPanel, "Center");

      if (lv_dtd == null) {
            lv_stuff = "Error reading DTD for " + lv_fragment;
            lv_currentPanel.add((new JLabel(lv_stuff, SwingConstants.LEFT)),
"Center");
      }
      else {
            // get all the UNIVERSAL elements and add to UI
            // (ie, those under the top level element.
            // not under SPECIAL, SYSTEM or CLIENT tags)
            ElementDecl lv_dtdElement        = DTDUtil.getRootElement(lv_dtd);
            TXElement   lv_docElement        = lv_fragment.getRootElement();
            ContentModel lv_contentModel     =
lv_dtd.getContentModel(lv_dtdElement.getName());
            CMNode       lv_contentModelNode =
lv_contentModel.getContentModelNode();

            TXDocument  lv_document         = lv_fragment.iv_document;
            //
            // ensureModelInDoc will make sure that all DTD model elements are
in document Root with default values
            // this will add the SYSTEM, CLIENT, all universal tags, and
special children elements
            createInterfaceForModel(lv_dtd, lv_fragment, lv_currentInputPanel,
lv_contentModelNode, lv_document, lv_docElement);
            lv_currentInputPanel.add(Box.createVerticalGlue());
            System.out.println("");

            // get all the SPECIAL elements and add to UI
            //lv_DTDElement = DTDUtil.getNamedElement(lv_dtd,
FranklinEditor.cv_SPECIAL_ELEMENT);
            //lv_docElement =
lv_fragment.getNamedElement(FranklinEditor.cv_SPECIAL_ELEMENT);
            //lv_contentModelNode = getContentModelNode(lv_dtd,
lv_DTDElement.getNodeName());
            //createInterfaceForModel(lv_dtd, lv_currentInputPanel,
lv_contentModelNode, lv_docElement);

```

```
625         }
626         return lv_currentPanel;
627   }
628   /**
629    * get the real component that is where the content is stored in the ui
630    * get the widget so we can update the elment in the DOM
631    * @return javax.swing.JComponent
632    * @param lv_component javax.swing.JComponent
633    */
634   public static JComponent getInterfaceComponent(JComponent lv_component) {
635         if (lv_component == null) return null;
636         if (lv_component instanceof JComboBox) return lv_component;
637         else {
638               Component[] lv_components = lv_component.getComponents();
639               return (JComponent)lv_components[0];
640         }
641   }
642   /**
643    * return the More or Less widget for this element
644    * @return javax.swing.JPanel
645    * @param lv_label java.lang.String
646    */
647   public static JPanel getMoreOrLess(String lv_label) {
648         return null;
649   }
650   /**
651    * This method was created in VisualAge.
652    */
653   public void newMethod() {
654         /*
655               Enumeration lv_enum = lv_dtd.getElementDeclarations();
656               while (lv_enum.hasMoreElements()) {
657                     ElementDecl lv_ed   = (ElementDecl)lv_enum.nextElement();
658                     lv_allElements.addElement(lv_ed.getName());
659                     FranklinEditor.printDebug("  " + lv_ed.getName());
660                     Enumeration lv_enum2 =
661   lv_dtd.getAttributeDeclarations(lv_ed.getName());
662                     while (lv_enum2.hasMoreElements()) {
663                           AttDef lv_attribute = (AttDef)lv_enum2.nextElement();
664                           String lv_attributeName = lv_attribute.getName();
665                           FranklinEditor.printDebug("      " + lv_attributeName);
666                     }
667               }
668
669               // hashtable of all elemnets and insertable elements for each
670   element
671               lv_hash = lv_dtd.prepareTable((String)lv_allElements.elementAt(0));
672               lv_enum = lv_allElements.elements();
673               while (lv_enum.hasMoreElements()) {
674                     lv_name = (String)lv_enum.nextElement();
675                     lv_hash.put(lv_name, new InsertableElement(lv_name));
676               }
677               */
678
679
680
681
```

```
682             /*
683   int lv_type = lv_contentModel.getType();
684   switch (lv_type) {                    // this case statement borrowed from
685   ContentModel.toString()
686         case ElementDecl.EMPTY :
687               // "EMPTY";
688               System.out.println("ContentType of " + lv_contentModelNode + " is
689   EMPTY");
690               break;
691         case ElementDecl.ANY :
692               // "ANY";
693               System.out.println("ContentType of " + lv_contentModelNode + " is
694   ANY");
695               break;
696         case ElementDecl.MODEL_GROUP :
697               //case ElementDecl.PCDATA:
698               if ((lv_contentModelNode instanceof CM1op) && ((CM1op)
699   lv_contentModelNode).getNode() instanceof CMLeaf) {
700                     CM1op cs1 = (CM1op) lv_contentModelNode;
701                     //ret = "(" + cs1.getNode() + ")" + (char)cs1.getType();
702                     System.out.println("ContentType 1 of " + lv_elementName + "
703   is (" + cs1.getNode() + ")" + (char) cs1.getType());
704               }
705               else if (lv_contentModelNode instanceof CMLeaf) {
706                     //ret = "(" + lv_contentModelNode + ")";
707                     System.out.println("ContentType 2 of " + lv_elementName + "
708   is (" + lv_contentModelNode + ")");
709               }
710               else
711
712                     //ret = this.modelGroupNode.toString();
713                     // getType on CM2op returns "|", or ","
714                     System.out.println("ContentType 3 of " + lv_elementName + "
715   is " + lv_contentModelNode.toString());
716                     System.out.println("            left " + ((CM2op)
717   lv_contentModelNode).getLeft());
718                     System.out.println("           right " + ((CM2op)
719   lv_contentModelNode).getRight());
720                     break;
721         }
722   */
723
724
725         /*
726         switch (lv_dtd.getContentType("TITLE")) {
727               case -1:
728                     // element is not declared
729                     System.out.println("getWidget: element " + lv_elementName + "
730   is not declared");
731                     break;
732               case ElementDecl.EMPTY:
733                     // any element is not insertable
734                     System.out.println("getWidget: any element is not insertable
735   to " + lv_elementName);
736                     break;
737               case ElementDecl.ANY:
738                     // Any element is insertable
```

```
739                     System.out.println("getWidget: any element is insertable to "
740 + lv_elementName);
741                     break;
742               case ElementDecl.MODEL_GROUP:
743                     // continued
744                     System.out.println("getWidget: Model_Group " +
745 lv_elementName);
746                     break;
747         }
748         */
749
750 }
751 /**
752  * Insert the method's description here.
753  * Creation date: (10/11/99 1:31:41 PM)
754  */
755 public static void printNode(CMNode lv_node) {
756       //String lv_elementName = lv_element.getName();
757       //ContentModel lv_contentModel = lv_dtd.getContentModel(lv_elementName);
758       //CMNode lv_contentModelNode = lv_contentModel.getContentModelNode(); //
759 if null, not a MODEL_GROUP
760
761       if (lv_node == null) { // not a MODEL_GROUP, ie ANY or EMPTY
762             System.out.println("printNode: not a model_group " + lv_node);
763       }
764       else {
765             //System.out.println("printNode: " + lv_node);
766             //case ElementDecl.PCDATA:
767 // has qualifiers * + or ?
768             if ((lv_node instanceof CM1op) &&
769                   ((CM1op)lv_node).getNode() instanceof CMLeaf) {
770                   CM1op lv_cs1 = (CM1op)lv_node;
771                   //ret = "(" + cs1.getNode() + ")" + (char)cs1.getType();
772                   System.out.println("ContentType 1 is (" + lv_cs1.getNode() +
773 ")" +
774                                                 (char) lv_cs1.getType());
775             }
776             else if (lv_node instanceof CMLeaf)
777 {                                 // no qualifiers *, + or ?
778                         //ret = "(" + lv_contentModelNode + ")";
779                         System.out.println("ContentType 2 is (" + lv_node +
780 ")");
781             }
782             else {
783                   //ret = this.modelGroupNode.toString();
784                   // getType on CM2op returns "|", or ","
785                   //System.out.println("ContentType 3 is " +
786 lv_node.toString());
787                   //System.out.println("            left " + ((CM2op)
788 lv_node).getLeft());
789                   //System.out.println("           right " + ((CM2op)
790 lv_node).getRight());
791                   printNode(((CM2op)lv_node).getLeft());
792                   printNode(((CM2op)lv_node).getRight());
793             }
794       }
795
```

```
/*
int lv_type = lv_contentModel.getType();
switch (lv_type) {                    // this case statement borrowed from
ContentModel.toString()
      case ElementDecl.EMPTY :
            // "EMPTY";
            System.out.println("ContentType of " + lv_contentModelNode + " is
EMPTY");
            break;
      case ElementDecl.ANY :
            // "ANY";
            System.out.println("ContentType of " + lv_contentModelNode + " is
ANY");
            break;
      case ElementDecl.MODEL_GROUP :
            //case ElementDecl.PCDATA:
            if ((lv_contentModelNode instanceof CM1op) && ((CM1op)
lv_contentModelNode).getNode() instanceof CMLeaf) {
                  CM1op cs1 = (CM1op) lv_contentModelNode;
                  //ret = "(" + cs1.getNode() + ")" + (char)cs1.getType();
                  System.out.println("ContentType 1 of " + lv_elementName + "
is (" + cs1.getNode() + ")" + (char) cs1.getType());
            }
            else if (lv_contentModelNode instanceof CMLeaf) {
                  //ret = "(" + lv_contentModelNode + ")";
                  System.out.println("ContentType 2 of " + lv_elementName + "
is (" + lv_contentModelNode + ")");
            }
            else

                  //ret = this.modelGroupNode.toString();
                  // getType on CM2op returns "|", or ","
                  System.out.println("ContentType 3 of " + lv_elementName + "
is " + lv_contentModelNode.toString());
                  System.out.println("            left " + ((CM2op)
lv_contentModelNode).getLeft());
                  System.out.println("           right " + ((CM2op)
lv_contentModelNode).getRight());
                  break;
      }
*/
}
/**
 * when processing the DTD elements, check the list to see if we need to
process it
 * in order to create a UI widget.
 * Creation date: (10/11/99 1:31:41 PM)
 */
public static boolean skipElement(String lv_elementName) {
      for (int i = 0; i < FranklinEditor.cv_skipElements.length; i++) {
            if (lv_elementName.equals(FranklinEditor.cv_skipElements[i]))
return true;
      }
      return false;
}
/**
```

```
852    * when processing the DTD elements, check the list to see if we need to
853   process this element
854    * in order to create a UI widget. here, we allow for processing our children
855   if we have them.
856    * Creation date: (10/11/99 1:31:41 PM)
857    */
858   public static boolean skipNonDisplayElement(String lv_elementName) {
859         for (int i = 0; i < FranklinEditor.cv_nonDisplayElements.length; i++) {
860               if (lv_elementName.equals(FranklinEditor.cv_nonDisplayElements[i]))
861   return true;
862         }
863         return false;
864   }
865   }
```